UC-Secure Protocols using Smooth Projective Hash Functions

Thèse d'habilitation

présentée et soutenue publiquement le 11 décembre 2017

pour l'obtention du

Diplôme d'Habilitation à Diriger des Recherches
de l'École normale supérieure
(spécialité Informatique)

Céline Chevalier
Maître de conférences à l'Université Panthéon-Assas, Paris II

Composition du jury :

Correspondant HDR :	David Pointcheval	(CNRS, DI/ENS, PSL Research University)
Rapporteurs :	Dennis Hofheinz	(Karlsruher Institut für Technologie)
	Louis Goubin	(Université de Versailles Saint-Quentin-en-Yvelines)
	Duong-Hieu Phan	(Université de Limoges, XLim)
Examinateurs :	Michel Abdalla	(CNRS, DI/ENS, PSL Research University)
	Stéphanie Delaune	(Univ Rennes, CNRS, IRISA)
	Fabien Laguillaumie	(Université Lyon 1)
	Damien Vergnaud	(Université Pierre et Marie Curie, Institut Universitaire de France)

Travaux effectués au Laboratoire d'Informatique de l'École normale supérieure
45 rue d'Ulm, 75230 Paris Cedex 05

À Lucas et Gabriel

Remerciements

L'heure est venue d'écrire cette fameuse page, qui occupe traditionnellement l'assemblée en attendant que l'exposé ne commence (ou que les questions se terminent enfin et que le pot commence). Pour ma thèse j'avais écrit que c'était la plus difficile à écrire : si j'avais su ce qui m'attendait huit ans plus tard... Écrire son mémoire d'habilitation, c'est avoir un peu l'impression qu'une page se tourne, même si une nouvelle page blanche se présente. S'il y a bien quelque chose qui n'a pas changé, c'est la crainte d'oublier des gens, ceux qui ont eu de l'importance ou un peu moins, ceux qui sont partis, ceux qui sont restés ou revenus. Je ne vais pas essayer d'être exhaustive, même si je m'excuse par avance en cas d'oubli. L'avantage de soutenir en même temps que de nombreux thésards en cette fin d'année, c'est que la majorité des personnes seront déjà citées quinze fois dans les autres manuscrits et ne m'en voudront donc pas trop. Dans le cas contraire, je compte sur toi Gaby, un sourire de ta part et tout sera oublié.

Mes remerciements les plus évidents s'adressent bien naturellement à David, qui a finalement réussi à me convaincre de soutenir cette HdR, qui m'a encouragée et soutenue dans cette entreprise, allant même jusqu'à m'envoyer un mail qui m'a bien fait rire début mai. C'est lui également qui m'a convaincue de rédiger pendant mon congé maternité (ce qui s'est révélé être l'un de ses pires conseils de ces dix dernières années), et qui a fait preuve d'énormément de patience envers moi. Mais bien au-delà de cette rédaction d'HdR passablement rocambolesque, je le remercie également pour tous ses conseils depuis tant d'années déjà (j'en écoute un grand nombre quand même...), et de m'avoir réaccueillie dans son équipe tout en essayant de me faire prendre mon envol avec plus ou moins de succès. J'ai beaucoup de chance de pouvoir profiter des conditions exceptionnelles de recherche à l'ENS et je lui en suis très reconnaissante.

Je remercie particulièrement Dennis, Louis et Hieu d'avoir accepté de rapporter ce mémoire, et Fabien, Stéphanie, Michel et Damien d'avoir accepté de faire partie de mon jury. J'adresse un très grand merci à mes coauteurs, en particulier Olivier, qui partage ses brillantes idées avec moi et supporte patiemment mes manies de formalisme et de notations et mon manque fréquent de disponibilité. Merci aussi à Damien d'avoir partagé son bureau avec moi pendant quelques années avant de nous quitter pour de plus hautes destinées. Toutes mes félicitations au passage ! Nos discussions sur nos petits et les photos de Louis sur ton bureau vont me manquer. Merci à tous les deux pour nos discussions sur Gtalk qui font souvent oublier la distance. Merci également à Hoeteck et David de m'avoir confié la responsabilité du projet EnBiD, qui une fois passé le stress de m'en occuper m'a finalement apporté beaucoup, et merci aussi pour l'invitation de Iordanis et la participation à la Winter School à Darmstadt, qui ont tout déclenché. Merci à Michele, Marc, Elham, Luka, Olivier et André, qui m'ont suivie dans cette aventure qui nous a conduit à écrire un projet ANR dans des délais assez déraisonnables. Merci à Michel de m'avoir suggéré cette idée l'an dernier.

Je remercie aussi celles et ceux qui font de l'ENS un endroit que je n'ai jamais eu envie de quitter, de nombreuses années plus tard : les membres permanents, thésards et post-docs pour la qualité de la recherche dans une bonne ambiance, le service informatique et l'équipe administrative, en particulier Joëlle, Sophie, Lise-Marie et Valérie pour leur efficacité, leur gentillesse et leur patience.

Je tiens également à remercier David N de m'avoir fait confiance et proposé de candidater à Paris 2. Après des années dans le microcosme de la prépa puis des ENS, il m'a fait découvrir une université surprenante mais sympathique dans laquelle je me sens bien. Merci aussi à mes collègues de Paris 2 ; je suis ravie de faire partie d'une équipe de maths aussi sympathique et de m'initier à la recherche en économie tout en ayant la liberté de poursuivre également mes recherches à l'ENS.

Je souhaite remercier Marta, les amis d'Olivier devenus les miens, Seb et les amis d'H&K, ceux qui le sont restés malgré la distance et le temps. Merci à Anne aussi, même si nos discussions ont moins rythmé ces derniers travaux qu'elles ne l'avaient fait pendant ma thèse. Merci à Michèle, Keren-Or et Michel, même si je ne peux pas venir aussi souvent que je le voudrais.

Enfin un immense merci à ma famille d'être là depuis toujours, Maman en premier lieu, bien sûr. À « Mamie » et « Tonton », qui s'occupent à merveille de Lulu et Gaby quand je suis à Paris, à Odile qui s'en occupe aujourd'hui pour que Maman puisse être un peu tranquille (si toutefois Lucas accepte de partager sa mamie...) Merci au dévouement de Maman et Nicolas qui nous ont fait une si jolie maison, et merci à Nicolas et Valérie d'être là et de supporter leur affreuse petite sœur avec tous ses défauts. Je m'excuse envers vous d'avoir rédigé en anglais, j'espère me rattraper avec la présentation en français.

Et bien sûr, un immense merci également à mes deux rayons de soleil qui sont trop petits pour lire ça. Merci p'tit Lu, d'être simplement trop mignon. Tu es la plus adorable crapule que je connaisse (en toute objectivité bien sûr). Et toi p'tit Gab, merci pour ta bouille craquante, ta bonne humeur permanente et tes sourires. Même si parfois le temps et la qualité de mon travail pâtissent un peu (beaucoup) de votre présence, je ne vous changerais pour rien au monde. Et le dernier merci va tout naturellement à Olivier, leur merveilleux papa.

Contents

Part I

Introduction

Chapter 1

Introduction

In this work, we study well-known two-party protocols, mainly PAKE *(Password-Authenticated Key Exchange)* and OT *(Oblivious Transfer)*, in a powerful security model called UC *(Universal Composability)* framework. As we will see, the constructions achieved heavily rely on a primitive called SPHF *(Smooth Projective Hash Function)*, that we extensively study in the following, combined with another primitive called *Commitment*.

1.1 Primitives and Protocols Studied

Oblivious Transfer (OT) is a notion introduced in 1981 by Rabin [Rab81]. In its classical 1-out-of-k version, it allows a receiver $\mathcal{R}$ to access a single line of a database while interacting with the server $\mathcal{S}$ owning the database containing k lines of data. In such schemes, the receiver should be oblivious to the other line values, while the server should be oblivious to which line was indeed received. Oblivious transfer has a fundamental role for achieving secure multi-party computation: It is for example needed for every bit of input in Yao's protocol [Yao86] as well as for Oblivious RAM ([WHC$^+$14] for instance), for every AND gate in the Boolean circuit computing the function in [GMW87] or for almost all known garbled circuits [BHR12]. It has been widely used and studied in the community, especially since a fundamental result by Kilian [Kil88], stating that one can achieve any multi-party computation scheme from oblivious transfer.

In the same vein, *Private Information Retrieval* (PIR) schemes [CGKS95] allow a user to retrieve information from a database, while ensuring that the database does not learn which data were retrieved. With the increasing need for user privacy, these schemes are quite useful in practice, be they used for accessing records for email repositories, collection of webpages, music... But while protecting the privacy of the user, it is equally important that the user should not learn more information than he is allowed to. This is called database privacy and the corresponding protocol is called a *Symmetrically Private Information Retrieval* (SPIR), which could be employed in practice, for medical data or biometric information. This notion is closely related to Oblivious Transfer.

Authenticated Key Exchange is quite an important notion for practical applications, since it enables two parties to generate a shared high entropy secret key. This secret key will be later used with symmetric protocols in order to protect communication, while interacting over an insecure network under the control of an adversary. Various authentication means have been proposed, and the most practical one is definitely a shared low entropy secret (called password) that the users can agree on over the phone. This kind of protocol, named PAKE, for *Password-Authenticated Key Exchange*, were proposed in 1992 by Bellovin and Merritt [BM92]. The small entropy space from which the password is drawn is subject to exhaustive search. The two participants, owning the same password, should end with the same private session key after the interaction. Due to their importance in practice, many schemes have been proposed and studied since then.

The Commit and Hash Paradigm. Many methods have been proposed in the past 20 years in order to construct such schemes, and moreover to prove them secure. First focusing on PAKE protocols, we choose to follow the idea introduced by Katz, Ostrovsky and Yung in [KOY01] and formally proven by

Gennaro and Lindell in [GL03] using the idea of *Hash Proof Systems* (or equivalently, *Smooth Projective Hash Functions*), designed by Cramer and Shoup in [CS02]. This idea roughly consists in *committing* to the password and then using the hashing primitive to generate the key, with specific properties so that the participants will obtain the same key if and only if they own the same password (and the computations were sound and done honestly). What we then remark is that the same idea can be applied to OT protocols.

Commitment schemes are two-party primitives (between a committer and a receiver) divided into two phases. In a first *commit* phase, the committer gives the receiver an analogue of a sealed envelope containing a value m, while in the second *opening* phase, the committer reveals m in such a way that the receiver can verify that it was indeed m which was contained in the envelope. It is required that a committer cannot change the committed value (i.e. he should not be able to open to a value different from the one he committed to), this is called the *binding* property. It is also required that the receiver cannot learn anything about m before the opening phase, this is called the *hiding* property. It is impossible to perfectly achieve both properties (rather than computationally or statistically) at the same time. El Gamal [ElG84] or Cramer-Shoup [CS02] encryptions are famous examples of perfectly binding commitments, and Pedersen encryption [Ped92] is the most known example of perfectly hiding commitments.

In our specific case of application to password-based authenticated key-exchange, in which the committed value is a password, one does not want to reveal this secret value during the opening phase. Instead, one wants the decommitment to be *implicit*, which means that the committer does not really open its commitment, but rather convinces the receiver that it actually committed to the value it pretended to. In the articles formerly cited, the authors achieve this property thanks to the notion of *Smooth Projective Hash Functions*.

Smooth Projective Hash Functions have been initially defined by Cramer and Shoup [CS02] and their application to PAKE schemes (among others) has been seen by Katz, Ostrovsky and Yung in [KOY01] and formally proven by Gennaro and Lindell in [GL03]. These hash functions are defined such that their value can be computed in two different ways if the input belongs to a particular subset (called the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language. The hash value obtained is indistinguishable from random in case the input does not belong to the language (property of *smoothness*) and in case the input does belong to the language but no witness is known (property of *pseudo-randomness*).

Applying this to PAKE schemes, when one considers the language of "well-formed commitments of the password owned by the other participant" (using the randomness as a witness), this ensures the implicit decommitment since the hash value will only be the same if the intended password is indeed the good one and the other participant indeed owns the witness (i.e. computed honestly the commitment on the aforesaid password).

Universal Composability Framework. In a traditional security model, this paradigm quite easily ensures the security. But an additional difficulty arises when one wants to prove the protocols in the universal composability framework proposed in [Can01]. In a nutshell, in the UC framework, security for a specific kind of protocol is captured by an ideal functionality (in an ideal world). A protocol is then proven secure if, given any adversary to the protocol in the real world, one can construct a simulator of this adversary in the ideal world, such that no environment can distinguish between the execution in the ideal world (between dummy players, the ideal functionality and the simulator of the adversary) and the execution in the real world (between the real players executing the real protocol and interacting between themselves and the adversary) in a non-negligible way.

Skipping the details, when the protocol makes use of commitments, this usually forces those commitments to be both *extractable* (meaning that a simulator can recover the value m committed to thanks to a trapdoor) and *equivocable* (meaning that a simulator can open a commitment to a value m' different from the value m it committed to thanks to another trapdoor), which is quite a difficult goal to achieve.

The now classical way [CF01, ACP09], **[ABB$^+$13]** to achieve both extractability and equivocability is to combine an equivocable CPA encryption scheme (such as Pedersen [Ped92]) and an extractable CCA encryption scheme (such as Cramer-Shoup [CS02]). As explained above, one then links them with an SPHF in order to obtain an implicit decommitment.

1.2 Related Work

Commitments. The first UC-secure commitment schemes were given by [CF01] and [DN02] and the authors of the former were the first to formalize the methodology combining an equivocable primitive and an extractable primitive. Many constructions have been proposed since then, for instance [Lin11a] and [FLM11] for the UC-commitment schemes and [KV11] for the UC PAKE schemes, in which the relations between commitments and SPHF have proven very useful.

Smooth Projective Hash Functions and Password-Authenticated Key-Exchange. These subjects cannot really be studied separately anymore, since SPHFs have been extensively used for PAKE schemes, which in return led to big improvements on the SPHF constructions.

The most famous instantiation of PAKE has been proposed by Bellovin and Merritt [BM92], and is called EKE, for Encrypted Key Exchange. It simply consists of a Diffie-Hellman key exchange [DH76], where the flows are symmetrically encrypted under the shared password. Overall, the equivalent of 2 group elements have to be sent.

A first formal security model was proposed by Bellare, Pointcheval and Rogaway [BPR00] (the BPR model), to deal with off-line dictionary attacks. It essentially says that the best attack should be the on-line exhaustive search, consisting in trying all the passwords by successive executions of the protocol with the server. Several variants of EKE with BPR-security proofs have been proposed in the ideal-cipher model or the random-oracle model [Poi12]. Katz, Ostrovsky and Yung [KOY01] proposed the first practical scheme (KOY), provably secure in the standard model under the DDH assumption. This is a 3-flow protocol, with the client sending 5 group elements plus a verification key and a signature, for a one-time signature scheme, and the server sending 5 group elements. It has been generalized by Gennaro and Lindell [GL03], who extended the initial definition of SPHF for an application to PAKE.

Their approach has thereafter been adapted to the *Universal Composability* (UC) framework by Canetti *et al.* [CHK$^+$05], who also proposed the first ideal functionality for PAKE protocols in the UC framework. Though quite efficient, their protocol was only secure against static corruptions (as opposed to adaptive adversaries, that are capable of corrupting players at any time, and learn their internal states). The first ones to propose an adaptively secure PAKE in the UC framework were Barak *et al.* [BCL$^+$05] using general techniques from multi-party computation. Though conceptually simple, their solution results in quite inefficient schemes. Building on the idea from [CF01], we improved the scheme from [CHK$^+$05] in [**ACP09**] to resist to adaptive adversaries, still in the standard model (without random oracles).

More recently, a variant of SPHF (called KV-SPHF in the following, as opposed to traditional CS-SPHF of Cramer-Soup and GL-SPHF of Gennaro-Lindell) proposed by Katz and Vaikuntanathan even allowed the construction of one-round UC-secure PAKE schemes [KV09, KV11], where the two players just have to send simultaneous flows to each other. In these SPHFs, the projection key depends on the hashing key only (and not on the word in the language), and the smoothness holds even if the word is chosen after having seen the projection key.

Oblivious Transfer. Since the original paper [Rab81], several instantiations and optimizations of OT protocols have appeared in the literature [NP01, CLOS02], including proposals in the UC framework. More recently, new instantiations have been proposed, trying to reach round-optimality [HK07], and/or low communication costs [PVW08]. Choi *et al.* [CKWZ13] proposed a generic method and an efficient instantiation secure against adaptive corruptions in the CRS model with erasures, but it is only 1-out-of-2 and it does not scale to 1-out-of-n OT, for $n > 2$. As far as adaptive versions of those protocols are concerned, this problem was first studied by [NP97, GH07, KNP11], and more recently UC secure instantiations were proposed, but unfortunately either under the Random Oracle, or under not so standard assumptions such as q-Hidden LRSW or later on q-SDH [CNs07, JL09a, RKP09, CDH12, GD14], but without allowing adaptive corruptions.

1.3 Our contributions

We focus here on our five more relevant publications and describe the results achieved in these papers. We also give the related results achieved in other papers when it is worth mentioning them.

1.3.1 Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages (PKC 2013, [BBC$^+$13a])

At PKC 2013 (see the full version page 95 or on eprint), with Fabrice Ben Hamouda, Olivier Blazy, David Pointcheval and Damien Vergnaud, we propose a new primitive that encompasses most of the previous notions of authenticated key exchange. It is closely related to *Credential-Authenticated Key Exchange* (CAKE) [CCGS10] and we call it LAKE, for *Language-Authenticated Key-Exchange*, since parties establish a common key if and only if they hold credentials that belong to specific (and possibly independent) languages. The definition of the primitive is more practice-oriented than the definition of CAKE but the two notions are very similar. In particular, the new primitive enables privacy-preserving authentication and key exchange protocols by allowing two members of the same group to secretly and privately authenticate each other without revealing this group beforehand.

In order to define the security of this primitive, we use the UC framework and an appropriate definition for languages that permits to dissociate the public part of the policy, the private common information the users want to check and the (possibly independent) secret values each user owns that assess the membership to the languages. We provide an ideal functionality for LAKE and give efficient realizations of the new primitive (for a large family of languages), that are secure under classical mild assumptions, in the standard model (with a common reference string – CRS), with static corruptions.

We significantly improve the efficiency of several CAKE protocols for specific languages and we enlarge the set of languages for which we can construct practical schemes. Notably, we obtain a very practical realization of Secret Handshakes and a Verifier-based Password-Authenticated Key Exchange.

1.3.2 New Techniques for SPHFs and Efficient One-Round PAKE Protocols (Crypto 2013, [BBC$^+$13b])

At Crypto 2013 (see the full version page 137 or on eprint), with Fabrice Ben Hamouda, Olivier Blazy, David Pointcheval and Damien Vergnaud, our first contribution is the description of an instantiation of KV-SPHF on Cramer-Shoup ciphertexts, and thus the first KV-SPHF on an efficient CCA encryption scheme. We thereafter use it within Katz and Vaikuntanathan's framework for one-round PAKE [KV11], in the BPR security model. Our scheme just consists of 6 group elements in each direction under the DDH assumption (4 for the ciphertext, and 2 for the projection key). This has to be compared with the 20 group elements, or more, in the best constructions known before, which all need pairing-friendly groups and pairing computations, or with the KOY protocol [KOY01] that has a similar complexity but with three sequential flows.

We also present the first GL-SPHFs and KV-SPHFs able to handle multi-exponentiation equations without requiring pairings. Our new KV-SPHFs enable several efficient instantiations of one-round LAKE protocols [**BBC$^+$13a**]. Our above one-round PAKE scheme is actually a particular case of a more general one-round LAKE scheme, for which we provide a BPR-like security model and a security proof. Our general constructions also cover CAKE [CCGS10].

1.3.3 SPHF-Friendly Non-Interactive Commitments (Asiacrypt 2013, [ABB$^+$13])

At Asiacrypt 2013 (see the full version page 169 or on eprint), with Michel Abdalla, Fabrice Ben Hamouda, Olivier Blazy, and David Pointcheval, we first define the notion of SPHF-friendly E^2-commitment together with an instantiation. The new construction is inspired by the commitment schemes in [CF01, CLOS02], [**ACP09**]. Like the construction in [**ACP09**], it combines a variant of the Cramer-Shoup encryption scheme (as an extractable commitment scheme) and an equivocable commitment scheme to be able to simultaneously achieve both equivocability and extractability. However, unlike the construction in [**ACP09**], we rely on Haralambiev's perfectly hiding commitment [Har11, Section 4.1.4], instead of the Pedersen commitment [Ped92].

Since the opening value of Haralambiev's scheme is a group element that can be encrypted in one ElGamal-like ciphertext to allow extractability, this globally leads to a better communication and computational complexity for the commitment. The former is linear in $m \cdot \mathfrak{K}$, where m is the bit-length of the committed value and $\mathfrak{K}$, the security parameter. This is significantly better than our extractable commitment construction in [**ACP09**] which was linear in $m \cdot \mathfrak{K}^2$, but asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m. However, we point out the latter proposals in [FLM11] are not SPHF-friendly since they are not robust.

We then show that a labeled E^2-commitment satisfying stronger notions of equivocability and extractability is a non-interactive UC-secure commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and a single global CRS, and we apply this result to our new construction.

Second, we provide a generic construction of a one-round UC-secure PAKE from any SPHF-friendly commitment, verifying an additional property called strong pseudo-randomness. The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS. In addition to being the first one-round adaptively secure PAKE, our new scheme also enjoys a much better communication complexity than previous adaptively secure PAKE schemes. For instance, in comparison to the PAKE in [**ACP09**], which was in 2013 the most efficient adaptively secure PAKE, the new scheme gains a factor $\mathfrak{K}$ in the overall communication complexity, where $\mathfrak{K}$ is the security parameter. However, unlike our scheme in [**ACP09**], our new construction requires pairing-friendly groups.

Third, we provide a generic construction of a three-round UC-secure 1-out-of-k OT from any SPHF-friendly commitment. The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS. Besides decreasing the total number of rounds with respect to existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solution in 2013 [CKWZ13]. Moreover, our construction is more general and provides a solution for 1-out-of-k OT schemes while the solution in [CKWZ13] only works for $k = 2$.

1.3.4 Adaptive Oblivious Transfer and Generalization (Asiacrypt 2016, [BCG16])

Due to their huge interest in practice, it is important to achieve low communication on Oblivious Transfer protocols, reducing the gap between them and Private Information Retrieval schemes. In *adaptive* OT schemes, the server is only required to send a message linear in the size k of the database for the first line queried. All the subsequent communication should be in $o(k)$.

At Asiacrypt 2016 (see the full version page 207 or on eprint), with Olivier Blazy and Paul Germouty, we give the first round-optimal adaptive Oblivious Transfer protocol secure in the UC framework with adaptive corruptions under standard assumptions (MDDH) and assuming reliable erasures. Our protocol builds on top of our UC-secure OT scheme [**BC15**] and adds the adaptivity by applying the idea of [GH07], which is round-optimal and based on Blind IBE. Roughly speaking, the idea is that each line (entry) of the database is encrypted using an IBE, and each line number corresponds to an identity. To obtain the s-th entry, the receiver then asks the secret key for identity s to the sender. The additional UC security (against adaptive corruptions) requirement furthermore adds some technicalities on the Blind IBE and the commitment used. We show how to instantiate the needed building blocks using standard assumptions, using or extending various basic primitives in order to fit the MDDH framework introduced in [EHK$^+$13]. It is interesting to note that our resulting adaptive Oblivious Transfer scheme has an amortized complexity in $\mathcal{O}(\log k)$, which is similar to current Private Information Retrieval instantiations [KLL$^+$15], that have weaker security prerequisites.

We also propose our new notion, named OLBE *(Oblivious Language-Based Envelope)* and provide a security model by giving a UC ideal functionality. We show that this notion supersedes the classical asymmetric automated trust negotiation schemes such as Oblivious Transfer, Oblivious Signature-Based Envelope, or even Access Controlled Oblivious Transfer, Priced Oblivious Transfer and Conditional Oblivious Transfer, in which the access to each line of the database is hidden behind some possibly secret restriction, be it a credential, a price, or an access policy. We show how to choose the languages in order to obtain from our framework all the corresponding ideal functionalities, recovering the known ones (such as OT) and providing the new ones (such as OSBE). We then give a generic scheme fulfilling our ideal functionality, which directly gives generic constructions for the specific cases.

1.3.5 Structure-Preserving Smooth Projective Hashing (Asiacrypt 2016, [BC16])

At Asiacrypt 2016 (see the full version page 247 or on eprint), with Olivier Blazy, we propose the notion of structure-preserving Smooth Projective Hash Functions (SP-SPHF), where both words, witnesses and projection keys are group elements. This is similar to the notion of structure-preserving signatures requiring the message, the signature, and the public keys to be group elements. Furthermore, hash and projective hash computations are doable with simple pairing-product equations in the context of bilinear groups. This allows, for example, to build SPHFs that implicitly demonstrate the knowledge of a Groth-Sahai Proof (serving as a witness).

As an example, we show that the UC-commitment from [FLM11] (while not fitting with the methodology of traditional SPHF we use in [**ABB$^+$13**]), is compatible with SP-SPHF and can be used to build UC protocols. As a side contribution, we first generalize this commitment from DLin to the $k-\mathsf{MDDH}$ assumption from [EHK$^+$13]. The combination of this commitment and the associated SP-SPHF then enables us to give interesting applications, among which a construction of a three-round 1-out-of-k OT, UC-secure against adaptive adversaries assuming reliable erasures and a single global CRS, and an instantiation of a one-round PAKE UC-secure against adaptive adversaries assuming reliable erasures and a single global CRS under any $k-\mathsf{MDDH}$ assumption. Contrarily to most existing one-round adaptively secure PAKE, we show that our scheme enjoys a much better communication complexity while not leaking information about the length of the password used.

1.3.6 Other Contributions

In [BCPV13] (ACNS 2013), with Olivier Blazy, David Pointcheval and Damien Vergnaud, we detail a possible inconsistency on the *binding* property of Lindell's second commitment scheme [Lin11a]. In order to avoid the above concern, we propose a simple patch to Lindell's scheme making it secure against adaptive corruptions.

We also improve both schemes by noting that the committer encrypts the value m (encoded as a group element) using the Cramer-Shoup encryption scheme [CS98]. In the opening phase, he simply reveals the value m and uses a Σ protocol to give an interactive proof that the message is indeed the one encrypted in the ciphertext. In Lindell's schemes, the challenge in the Σ protocol is sent to the committer using a "dual encryption scheme". Our improvement consists in noting that the receiver can in fact send this challenge directly without having to send it encrypted before. With additional modifications of the schemes, we can present two new protocols secure under the DDH assumption in the UC framework, against static and adaptive corruptions. Both schemes requires a smaller bandwidth and less interactions than the original schemes.

In [BC15] (ACNS 2015), with Olivier Blazy, we show that we can broaden the class of primitives that can be used for the equivocable part of the SPHF-friendly commitments defined in [**ABB$^+$13**]. We give a generic construction of SPHF-friendly commitments from two conceptually easier building blocks: a collision-resistant chameleon hash (CH, introduced in [KR00]) function which is verifiable (either publicly or for the receiver only) and an SPHF-friendly CCA encryption scheme. The extra requirement on the CH function is simple to achieve as soon as only classical algebraic operations are applied to the randomness, and SPHF-friendly encryption is now well-known since [CS02], with several instances (contrary to SPHF-friendly commitments, which is a difficult task). We then give three instantiations of this SPHF-friendly scheme, respectively based on DDH, LWE and DCR.

While the UC-secure OT construction we give in [**ABB$^+$13**] is an ad hoc solution with pairings, we obtain here a generic construction (from CH and CCA encryption) which does not specifically induce pairings. Furthermore, our 3 instantiations come straightforward from our generic framework (and [**ABB$^+$13**] can be derived from it).

In [BCV15] (RSA 2015), with Olivier Blazy and Damien Vergnaud, we present an efficient non-interactive technique to prove (in zero-knowledge) that a committed message does not belong to a set $\mathcal{L}$. The proof is generic and relies on a proof of membership to $\mathcal{L}$ with specific mild properties. In particular, it is independent of the size of $\mathcal{L}$ and if there exists an efficient proof of membership for committed values, one gets readily an efficient proof of non-membership. Instantiated with a combination of SPHF and Groth-Sahai proof system, we obtain very efficient realization for non-interactive proof of non-membership of committed values.

In 2009, Kiayias and Zhou [KZ09] introduced *zero-knowledge proofs with witness elimination*. This primitive enables to prove that a committed message m belongs to a set L (with a witness w) in such a way that the verifier accepts the interaction only if w does not belong to a set determined by a public relation Q and some *private* input w' of the verifier. The verifier does not learn anything about w (except that $m \in \mathcal{L}$ and $(w, w') \notin$ Q) and the prover does not learn anything about w'. The primitive can obviously be used to handle revocation lists. We show that the original proposal of zero-knowledge proofs with witness elimination from [KZ09] is flawed and that a dishonest prover can actually make a verifier accept a proof for any message $m \in \mathrm{L}$ even if $(w, w') \in \mathrm{Q}$. In particular, in the suspect tracking scenario, a dishonest prover can identify himself even if he is on the suspect list. Therefore, their protocol does not achieve the claimed security. However we explain how to apply our proof of non-membership to fix it. We obtain a proof system that achieves the security goal and is more efficient than the original (insecure) solution.

Finally, we briefly present applications of our proof of non-membership to other settings such as anonymous credentials and privacy-preserving authenticated key exchange.

In [BCV16] (RSA 2016), with Olivier Blazy and Damien Vergnaud, we consider distributed PAKE, which was designed to prevent the adversary from recovering a whole database of passwords in case of server corruption. In this setting, the password database on the server side is shared among two servers, and authentication requires a distributed computation involving the client, only owning his password, and the two servers, who will use some additional shared secret information. The interaction is performed using a *gateway* that does not know any secret information and ends up in the gateway and the client sharing a common key.

We first define a security model, based on a variant of the BPR model in the distributed setting proposed by Katz, MacKenzie, Taban and Gligor in [KMTG12].

In our first construction, the user generates information theoretic shares of his password and sends them to the servers. In the authentication phase, the parties run a dedicated protocol to verify that the provided password equals the priorly shared one. This is achieved by combining six SPHFs. However, it implies that at each time period, the servers have to refresh the information-theoretic sharing of the password of all users.

In our second construction, passwords are now encrypted using a public-key encryption scheme where the corresponding secret key is shared among the servers. At the beginning of each time period, the servers only need to refresh the sharing of this secret key but the password database is not modified (and can actually be public). Password verification and the authenticated key exchange is then carried out without ever decrypting the database. A secure protocol is run to verify that the password sent by the user matches the encrypted password, again by combining several SPHFs.

In [BCG17] (ACNS 2017), with Olivier Blazy and Paul Germouty, we propose a generic transformation from Password-Authenticated Key Exchange to Oblivious Transfer (the reverse transformation was already studied in [CDVW12]). Our framework allows to transform a UC-secure PAKE into a UC-secure OT with the same level of security (for instance resistance to adaptive corruptions).

We choose to focus on the transformation of a 2-round PAKE to a 3-round OT since this kind of PAKE is the most commonly encountered and the most efficient one. Furthermore, this allows us to give a generic optimization of our transformation in this specific case, exploiting the fact that the server does not need to hide his "password", since his password is a (public) number of line in the database.

After showing an application of our technique on our PAKE scheme from [**ABB$^+$13**], which allows us to immediately recover the associated OT scheme given in this article, we show that our transformation also applies to the PAKE scheme from [JR15], allowing us to give a nearly optimal OT scheme.

Our technique works with every possible PAKE scheme, be they elliptic-curve-based or not. Furthermore, also note that it also allows to transform a BPR-secure PAKE [BPR00] into an OT secure in a game-based security model. In order to illustrate these last two points, we apply our framework to a BPR-secure lattice-based PAKE scheme proposed by [KV09], giving us a secure lattice-based OT scheme.

1.4 Outline

After a brief introduction to the UC framework in the rest of this part (Chapter 2 page 23), we present the cryptographic primitives in Part II (Chapters 3 and 4) and the cryptographic protocols in Part III (Chapters 5 and 6).

In Chapter 3 page 31, we start by giving the definition and classical properties of SPHFs. We then give an example of KV-SPHF construction (for Cramer-Shoup ciphertexts) that was originaly published in [**BBC$^+$13b**] and a quick summary of the constructions we were able to achieve in the different papers. We conclude by giving our new notion of structure-preserving SPHF [**BC16**] and a summary of its possible applications.

In Chapter 4 page 39, we start by giving the definition and classical properties of commitment schemes. We then describe our correction and improvement of Lindell's commitment scheme [Lin11a], [**BCPV13**]. We finish by extending the definitions of equivocability and extractability, and by defining the notion of SPHF-friendly commitments [**ABB$^+$13, BC15, BC16**] useful for the PAKE and OT schemes described later.

In Chapter 5 page 47, we start by giving the definition and ideal functionality for PAKE schemes. We then focus on one-round constructions [KV11], [**BBC$^+$13b, ABB$^+$13, BC16**], before describing the new notion of Language-Authenticated Key-Exchange (LAKE) [**BBC$^+$13a, BBC$^+$13b**], giving some applications. Finally, we describe the security model and constructions for Distributed PAKE [**BCV16**].

In Chapter 6 page 61, we start by giving the definition and ideal functionality for OT schemes. We then describe our three-round constructions [**ABB$^+$13, BC15, BC16**], before dealing with the adaptive version of the protocol [**BCG16**] with a security model and a practical instanciation. We finally describe our extension to Oblivious Language-Based Envelope (OLBE) [**BCG16**], before giving our transformation of a PAKE scheme into an OT scheme [**BCG17**].

Finally, the full version of the five articles described earlier can be found in Part V page 93.

1.5 Publications

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology – ASIACRYPT 2013, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 214–234, Bengalore, India, December 1–5, 2013. Springer, Heidelberg, Germany.

[ABCP09] Michel Abdalla, Xavier Boyen, Céline Chevalier, and David Pointcheval. Distributed public-key cryptography from weak secrets. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009: 12th International Conference on Theory and Practice of Public Key Cryptography*, volume 5443 of *Lecture Notes in Computer Science*, pages 139–159, Irvine, CA, USA, March 18–20, 2009. Springer, Heidelberg, Germany.

[ACCP08] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient two-party password-based key exchange protocols in the UC framework. In Tal Malkin, editor, *Topics in Cryptology – CT-RSA 2008*, volume 4964 of *Lecture Notes in Computer Science*, pages 335–351, San Francisco, CA, USA, April 7–11, 2008. Springer, Heidelberg, Germany.

[ACCP09] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Password-authenticated group key agreement with adaptive security and contributiveness. In Bart Preneel, editor, *AFRICACRYPT 09: 2nd International Conference on Cryptology in Africa*, volume 5580 of *Lecture Notes in Computer Science*, pages 254–271, Gammarth, Tunisia, June 21–25, 2009. Springer, Heidelberg, Germany.

[ACGP11] Michel Abdalla, Céline Chevalier, Louis Granboulan, and David Pointcheval. Contributory password-authenticated group key exchange with join capability. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 142–160, San Francisco, CA, USA, February 14–18, 2011. Springer, Heidelberg, Germany.

[ACMP10] Michel Abdalla, Céline Chevalier, Mark Manulis, and David Pointcheval. Flexible group key exchange with on-demand computation of subgroup keys. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10: 3rd International Conference on Cryptology in Africa*, volume 6055 of *Lecture Notes in Computer Science*, pages 351–368, Stellenbosch, South Africa, May 3–6, 2010. Springer, Heidelberg, Germany.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Heidelberg, Germany.

[BBC$^+$13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Theory and Practice of Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291, Nara, Japan, February 26 – March 1, 2013. Springer, Heidelberg, Germany.

[BBC$^+$13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15: 13th International Conference on Applied Cryptography and Network Security*, volume 9092 of *Lecture Notes in Computer Science*, pages 65–86, New York, NY, USA, June 2–5, 2015. Springer, Heidelberg, Germany.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 339–369, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[BCFP10] Xavier Boyen, Céline Chevalier, Georg Fuchsbauer, and David Pointcheval. Strong cryptography from weak secrets. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10: 3rd International Conference on Cryptology in Africa*, volume 6055 of *Lecture Notes in Computer Science*, pages 297–315, Stellenbosch, South Africa, May 3–6, 2010. Springer, Heidelberg, Germany.

[BCG16] Olivier Blazy, Céline Chevalier, and Paul Germouty. Adaptive oblivious transfer and generalization. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 217–247, Hanoi, Vietnam, December 4–8, 2016. Springer, Heidelberg, Germany.

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost Optimal Oblivious Transfer from QA-NIZK. In *ACNS 2017*, LNCS. Springer-Verlag, to appear, 2017.

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13: 11th International Conference on Applied Cryptography and Network Security*, volume 7954 of *Lecture Notes in Computer Science*, pages 534–551, Banff, AB, Canada, June 25–28, 2013. Springer, Heidelberg, Germany.

[BCTV16] Fabrice Benhamouda, Céline Chevalier, Adrian Thillard, and Damien Vergnaud. Easing Coppersmith methods using analytic combinatorics: Applications to public-key cryptography with weak pseudorandomness. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016: 19th International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 9615 of *Lecture Notes in Computer Science*, pages 36–66, Taipei, Taiwan, March 6–9, 2016. Springer, Heidelberg, Germany.

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-interactive zero-knowledge proofs of non-membership. In Kaisa Nyberg, editor, *Topics in Cryptology – CT-RSA 2015*, volume 9048 of *Lecture Notes in Computer Science*, pages 145–164, San Francisco, CA, USA, April 20–24, 2015. Springer, Heidelberg, Germany.

[BCV16] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In Kazue Sako, editor, *Topics in Cryptology – CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 3–18, San Francisco, CA, USA, February 29 – March 4, 2016. Springer, Heidelberg, Germany.

[CC11] Hervé Chabanne and Céline Chevalier. *Vaudenay's Privacy Model in the Universal Composability Framework: A Case Study*, pages 16–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[CDK11] Céline Chevalier, Stéphanie Delaune, and Steve Kremer. Transforming password protocols to compose. In Supratik Chakraborty and Amit Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, volume 13 of *LIPIcs*, pages 204–216. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[CDKR13] Céline Chevalier, Stéphanie Delaune, Steve Kremer, and Mark Dermot Ryan. Composition of password-based protocols. *Formal Methods in System Design*, 43(3):369–413, 2013.

[CFPZ09] Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal randomness extraction from a Diffie-Hellman element. In Antoine Joux, editor, *Advances in Cryptology – EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 572–589, Cologne, Germany, April 26–30, 2009. Springer, Heidelberg, Germany.

[CLV16] Céline Chevalier, Fabien Laguillaumie, and Damien Vergnaud. Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016: 21st European Symposium on Research in Computer Security, Part I*, volume 9878 of *Lecture Notes in Computer Science*, pages 261–278, Heraklion, Greece, September 26–30, 2016. Springer, Heidelberg, Germany.

Chapter 2

Security Model

We give here a high-level description of the so-called UC (for *Universal Composability*) framework. It is a security model tailored for *multiparty computation*, in which n players interact in order to compute securely a given function of their inputs. *Securely* here means that the output is correct and that the secrecy of the inputs is insured, even if a certain number of players are cheating.

Formally, one considers n players P_i, each owning an input x_i, and an n-variable function f. The goal is to compute $f(x_1, \dots, x_n) = (y_1, \dots, y_n)$ such that each player P_i learns y_i and nothing more.

Yao gave in [Yao82] a good example, known as the "millionnaires' problem ". Imagine two millionnaires meeting in the street and willing to establish who is the richest among the two, without revealing the exact sum each one possesses. The considered function is here a comparison between two integers: each one will learn the result (who owns more money), but nothing more (and in particular not the amount of money owned by the other one). One can also think of voting systems: each voter has to learn the result of the vote, and be certain it is the good one, but not the name of the candidate for which its neighbour voted. These two examples are particular cases in which the y_i are equal, which is not always the case. In a blind signature scheme for instance [Cha82], useful in electronic payment systems, the signer gives its private signing key as input and learns nothing, whereas the user gives the message to be signed as input and learns the signature as output.

It is necessary to give a formal mathematical framework and a security model to prove thoroughly the security of such a protocol. In the case of multi-party protocols, this is not so easy, given the diversity of the possibilities.

A good definition necessarily has to deal correctly with particular cases (such as electronic vote): for instance, one can make a list of the required properties (secret input, correct result...), but it is difficult to be sure that such a list is exhaustive (for instance, the model has to ensure that a player cannot behave in a way related to the behaviours of its neighbours).

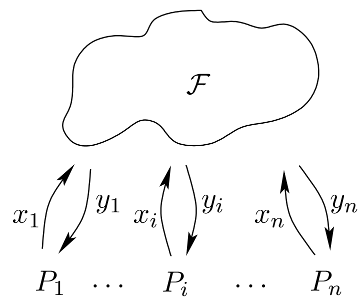
In order to solve this inherent problem, the inventor of the UC framework, Canetti [Can01] started from a completely different point of view, by considering for each protocol, the "ideal behaviour" it should have.

2.1 Description of the UC Framework

2.1.1 Ideal World and Real World

A protocol and the corresponding attacks lie in the so-called *real world.* On the opposite side, one considers an *ideal world*, in which everything would happen in a perfect way: This world describes the specifications of the protocol. The main idea behind the UC framework is then to say that a protocol is secure if its behaviour is indistinguishable from its ideal counterpart.

More formally, we define in the ideal world an entity that one can never corrupt, called the *ideal functionality* and usually denoted as $\mathcal{F}$. The players privately send their inputs to this entity, and receive their corresponding output the same way. There is no communication between the different players, as sketched in the following picture. $\mathcal{F}$ is assumed to behave in a perfectly correct way, without revealing information other than required, and without being possibly corrupted by an adversary. If it is well designed, we usually say that $\mathcal{F}$ is *trivially secure and correct.*



In the case of shared computation, the secure evaluation of a function can be specified by a trusted authority which receives all the inputs, computes the outputs, and gives back those values to the players. The main advantage of the UC framework is to allow the treatment of reactive tasks, by replacing this trusted party by the ideal functionality, which is a more general algorithmic entity.

Since the goal of this security model is to guarantee the robustness of a protocol with respect to the environment in which it is executed, one considers the protocols independently of the environment and tries to prove that it remains secure when composed with other protocols. One thus defines tasks for a unique instance, even if the protocol is supposed to be used with multiple and concurrent instances. This enables to have simple formulations of ideal functionalities and an automatic guarantee of the security of a protocol in a multi-task setting, thanks to a so-called *universal composability theorem*.

Such examples of basic tasks have been formalized with an adapted functionality ([Can01]): message authentication $\mathcal{F}_{\text{AUTH}}$, secure message transmission $\mathcal{F}_{\text{SMT}}$), secure communication sessions and *key exchange* $\mathcal{F}_{\text{SCS}}$), public-key encryption, digital signatures, coin-tossing, commitment, zero-knowledge, oblivious transfer...

Once $\mathcal{F}$ is defined, the goal of a protocol π, executed in a real world in the presence of an adversary, is then to create a situation equivalent to that obtained with $\mathcal{F}$. This means that the communication between players should not give more information than that given by the description of the function itself and the result of the latter (which can suffice to forbid any kind of security, for instance when considering the constant function).

2.1.2 Adversary and Environment

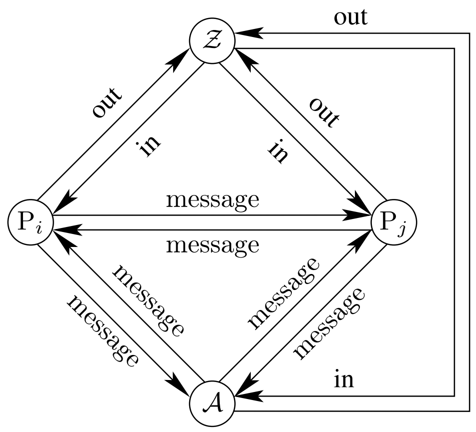
As always in cryptography, there exists an adversary which spies on the players and can do whatever it wants to interfere with the execution of the protocol. The adversary can either be *adaptive*, i.e. allowed to corrupt users whenever it likes to, or *static*, i.e. required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathcal{A}$ has complete access to the internal state and the private values of the player, takes its entire control, and plays on its behalf.

But in the UC framework, there also exists a second type of adversary, playing the role of the distinguisher, called the *environment*. As implied by its name, it models "everything that is outside the protocol being executed". The environment is the one giving the inputs to the players and recovering the outputs from them in the end of the execution. This models the fact that, in general, the inputs are not randomly chosen. Indeed, they rather depend on the "environment" (in its classical meaning) in which the protocol is executed. For instance, an email is most often an answer to another email.

As said before, in order to be secure, a protocol has to perfectly mimic its ideal counterpart: It is the environment which will have to judge that, by giving a single bit output, saying whether it thinks it interacted with the real or the ideal protocol. The goal of the UC framework is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by the ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary.

Since the adversary can interact at any moment with the environment, we say that the role of the latter is to be an interactive distinguisher. This gives a huge constraint to the simulator, which cannot "rewind" the environment, which is a classical trick in traditional security proofs. The main difference with classical proofs lies in the way the environment interacts with the adversary. In the UC framework, they can interact freely and anytime; This models the fact that the information between the considered instance and the remaining of the network can be exchanged anytime.

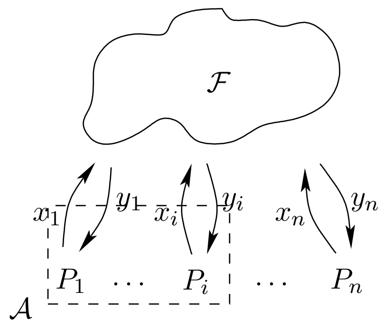
The environment can measure the influence of the adversary $\mathcal{A}$ on the outputs of the players, the leakage of information as well as the time for delivering the messages. The allowed interactions are drawn on the following picture. The adversay has access to the communication between players, but not to the inputs and outputs of the honest players (it completely controls the dishonest or corrupted players). On the contrary, the environment has access to the inputs and outputs of all players, but not to their communication, nor to the inputs and outputs of the subroutines they can invoque. Finally, it is aware of the identities of the players corrupted by the adversary.



2.1.3 Corruptions

In classical models, depending on the security framework, the privacy of the session key is either modeled via the "semantic security" [BR94] or via the indistinguishability between the protocol and its ideal counterpart [CK01]. In either case, the leakage of the long-term secret (such as a password) is modeled by "corruption" queries. Unfortunately, a corruption can reveal more information than the sole long-term secrets. Since the leakage of ephemeral secrets can cause important damages in several contexts [Kra05, KM06], one has to consider "strong corruptions" [Sho99]. However, it seems unrealistic to allow the designer of a protocol to decide which precise element is revealed or not by a corruption query.

On the contrary, the UC framework allows for a different approach: When a strong corruption occurs, the adversary gains access to the whole intern memory of the corrupted player (and thus learns its internal state) and then takes its entire control in order to act on its behalf for the remaining of the execution. This seems to be a much more realistic scenario. In a real execution of the protocol, this is modeled by giving the adversary the long-term secrets (such as the passwords) as well as the internal state of the corrupted player. Furthermore, the adversary can from now on arbitrarily modify the strategy of the player. In an ideal execution of the protocol, the simulator $\mathcal{S}$ learns the player's long-term secrets from the functionality and has to simulate its internal state in such a way that everything remains consistent with what the environment had given to the player at the beginning and everything that has followed since then.



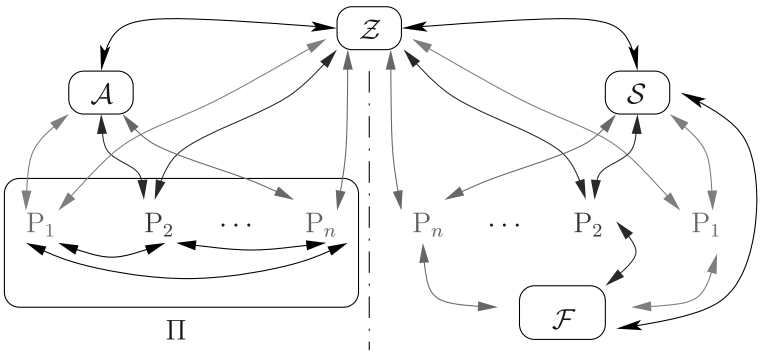
2.2 Main Ingredients for UC Security

2.2.1 Ideal Functionalities

In the UC framework, the security is thus defined in terms of an ideal functionality $\mathcal{F}$, which is a trusted authority interacting with a set of players in order to compute a given function f. In particular, the players give their inputs to $\mathcal{F}$, which then applies f to these values and returns as a result the corresponding value as output to each player. In this ideal framework, the security is thus inherently guaranteed, since an adversary controlling certain players can only learn (and maybe modify) the data of corrupted players.

More formally, in order to prove that a candidate protocol π UC-emulates the ideal functionality $\mathcal{F}$, one considers an environment $\mathcal{Z}$, which is allowed to give inputs to all players, and has to construct, for any polynomial adversary $\mathcal{A}$ (which controls the communication between the players), a *simulator* $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ (the *distinguisher*) can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players only interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage.

The UC theorem then implies that π keeps on behaving as the ideal functionality even if it is executed in an arbitrary environment.



2.2.2 Session and Player Identifiers

In the UC framework, there can be several copies of the ideal functionality running in parallel. Each one is then assumed to own a unique *session identifier* (SID). Each time a message is sent to a specific copy of this functionality $\mathcal{F}$, this message has to contain the SID of the copy to which it is intended. Following [CHK$^+$05], we assume that each protocol UC-emulating $\mathcal{F}$ expects inputs containing the proper SID.

2.2.3 UC Framework With Joint State

The universal composition theorem allows for the security analysis of a system seen as a simple entity, but it gives no information in case several protocols share a given number of states and randomness (as a secret key, for instance). In many cases, it thus cannot be used as is, since different sessions of a unique protocol share some data (such as a random oracle, an ideal cipher or a common reference string).

In order to overcome this issue, Canetti and Rabin introduced in 2003 the notion of *universal composability with joint state* [CR03]. Informally speaking, it consists in a new composition operation which enables several protocols to share some states while preserving security. This is done by defining the *multi-session extension* of $\mathcal{F}$, which runs in parallel multiple extensions of $\mathcal{F}$. Each copy of $\mathcal{F}$ owns a *subsession identifier* (SSID) which means that, if $\widehat{\mathcal{F}}$ receives a message m with SSID ssid, then it sends m to the copy of $\mathcal{F}$ owning the SSID ssid. If this copy does not exist, $\widehat{\mathcal{F}}$ invokes a new one on-the-go. To sum up, when $\widehat{\mathcal{F}}$ is executed, the protocol has to specify both the SID (the session identifier, as with any ideal functionality) and the intended SSID.

2.2.4 The Split Functionality

Without any strong authentication mechanisms, since the network is controlled by the adversary, the latter can always partition the players into disjoint subgroups and execute independent sessions of the protocol with each subgroup, playing the role of the other players. Such an attack is unavoidable since players cannot distinguish the case in which they interact with each other from the case where they interact with the adversary. We call such an attack an *implicit corruption*, which means that the adversary impersonates one or several players from the beginning, before the key generation. The authors of [BCL$^+$05] addressed this issue by proposing a new model based on *split functionalities* which guarantees that this attack is the only one available to the adversary. More generally, the authors have considered protocols for generic *multiparty computation* without authenticated channels. In particular, they give a general and conceptually simple (though inefficient) solution to a certain number of problems, including group password-based key exchange.

The split functionality is a generic construction based upon an ideal functionality: Its description can be found in Figure 2.1. In the initialization stage, the adversary $\mathcal{A}$ adaptively chooses disjoint subsets of the honest parties (with a unique session identifier that is fixed for the duration of the protocol). More precisely, the protocol starts with a session identifier sid. Then, the initialization step generates random values which, randomly combined and associated with sid, create the new session identifier sid', shared among all players which have received the same values – that is all the players of the disjoint subsets. The important point is that all the subsets create a *partition* of the players, thus forbidding any communication between the subsets. During the computation, each subset H activates a separate instance of the functionality $\mathcal{F}$. All these functionality instances are independent: The executions of the protocol for each subset H can only be related in the way $\mathcal{A}$ chooses the inputs of the players it controls. The parties $\mathrm{P}_i \in \mathrm{H}$ provide their own inputs and receive their own outputs, whereas $\mathcal{A}$ plays the role of all the parties $\mathrm{P}_j \notin \mathrm{H}$.

In the particular case of password-authenticated key exchange, the adversary then sends a $\mathsf{NewSession}$ query on behalf of such implicitly corrupted players, which never become really corrupted, but have always been controlled by the adversary. This situation is modeled in the ideal world by the split functionalities $s\mathcal{F}$, which make one or several calls to normal functionalities $\mathcal{F}$ on disjoint subsets of (real) players.

Fig. 2.1 – The Split Functionality $s\mathcal{F}$

Given a functionality $\mathcal{F}$, the split functionality $s\mathcal{F}$ proceeds as follows:

Initialization:

- Upon receiving (Init, sid) from party P_i, send $(\mathsf{Init}, sid, \mathrm{P}_i)$ to the adversary.
- Upon receiving a message $(\mathsf{Init}, sid, \mathrm{P}_i, \mathrm{H}, sid_\mathrm{H})$ from $\mathcal{A}$, where H is a set of party identities, check that P_i has already sent (Init, sid) and that for all recorded $(\mathrm{H}', sid_{\mathrm{H}'})$, either $\mathrm{H} = \mathrm{H}'$ and $sid_\mathrm{H} = sid_{\mathrm{H}'}$ or H and H′ are disjoint and $sid_\mathrm{H} \neq sid_{\mathrm{H}'}$. If so, record the pair $(\mathrm{H}, sid_\mathrm{H})$, send $(\mathsf{Init}, sid, sid_\mathrm{H})$ to P_i, and invoke a new functionality $(\mathcal{F}, sid_\mathrm{H})$ denoted as $\mathcal{F}_\mathrm{H}$ and with set of honest parties H.

Computation:

- Upon receiving (Input, sid, m) from party P_i, find the set H such that $\mathrm{P}_i \in \mathrm{H}$ and forward m to $\mathcal{F}_\mathrm{H}$.
- Upon receiving $(\mathsf{Input}, sid, \mathrm{P}_j, \mathrm{H}, m)$ from $\mathcal{A}$, such that $\mathrm{P}_j \notin \mathrm{H}$, forward m to $\mathcal{F}_\mathrm{H}$ as if coming from P_j.
- When $\mathcal{F}_\mathrm{H}$ generates an output m for party $\mathrm{P}_i \in \mathrm{H}$, send m to P_i. If the output is for $\mathrm{P}_j \notin \mathrm{H}$ or for the adversary, send m to the adversary.

2.2.5 A Simpler Model

Canetti, Cohen and Lindell formalized a simpler variant of the UC Framework in [CCL15]. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated (as if we worked in the $\mathcal{F}_{\mathrm{AUTH}}$-hybrid model); There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions. We refer the interested reader to [CCL15] for details.

2.3 Formalisation of Ideal Models

As for every primitive, ideal models have to be formalized in the universal composability framework by the corresponding ideal functionalities. For instance, one will speak of a protocol secure in the $\mathcal{F}_{\mathrm{RO}}$-hybrid model in order to speak of a protocol secure in the random oracle model.

Canetti *et al.* show in [CHK+05] the impossibility of UC-secure password-authenticated key exchange without any additional setup assumptions. This motivates the study of protocols secure in the hybrid models $\mathcal{F}_{\mathrm{RO}}$ and $\mathcal{F}_{\mathrm{CRS}}$, as described in this section.

2.3.1 Random Oracle in the UC Framework

The ideal functionality for a random oracle [BR93] has been defined by Hofheinz and Müller-Quade in [HMQ04]. We present it here in Figure 2.2 . It is clear that the random oracle model (ROM) UC-emulates this functionality.

Since the session identifier sid is included in the inputs to the random oracle in the protocols, one can use a single instantiation of the random oracle, lying in the joint state, rather than a different instantiation for each session, which is more realistic. Note however that in the cases where only part of the session identifier is included, this could lead to collisions with other sessions sharing this part of sid, and thus in issues during the simulation (and more precisely the programming of the oracle). But the oracle is only programmed for honest players. The only important point is thus to ensure, when designing a protocol, that this programming will only be needed only once (except, of course, if the adversary already asked the critical query, but this only happens with negligible probability). This will usually be done thanks to the split functionality, as described in Section 2.2.4.

Fig. 2.2 – The Functionality $\mathcal{F}_{\mathbf{RO}}$

The functionality $\mathcal{F}_{\mathrm{RO}}$ proceeds as follows, running on security parameter k, with parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$ and an adversary $\mathcal{S}$:

- $\mathcal{F}_{\mathrm{RO}}$ keeps a list $\mathrm{L}_{\mathsf{sid}}$ (which is initially empty) of pairs of bitstrings.
- Upon receiving a value (sid, m) (with $m \in \{0,1\}^*$) from some party P_i or from $\mathcal{S}$, do:
  - If there is a pair $(m, \tilde{h})$ for some $\tilde{h} \in \{0,1\}^k$ in the list $\mathrm{L}_{\mathsf{sid}}$, set $h := \tilde{h}$.
  - If there is no such pair, choose uniformly $h \in \{0,1\}^k$ and store the pair $(m, h) \in \mathrm{L}$.

  Once h is set, reply to the activating machine (i.e. either P_i or $\mathcal{S}$) with (sid, h).

2.3.2 The Common Reference String Model

The functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ was presented by Barak, Canetti, Nielsen and Pass in [BCNP04]. At each call of $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$, it sends back the same reference string, chosen by itself, following a known public distribution $\mathcal{D}$. We recall it here in Figure 2.3 .

Fig. 2.3 – The Functionality $\mathcal{F}_{\mathsf{CRS}}^{\mathcal{D}}$

The functionality $\mathcal{F}_{\mathrm{CRS}}^{\mathcal{D}}$ is parameterized by a distribution $\mathcal{D}$. It interacts with a set of players and an adversary in the following way:

- Choose a value $r \xleftarrow{\$} \mathcal{D}$.
- Upon receiving a value $(\mathsf{CRS}, \mathsf{sid})$ from a player, send $(\mathsf{CRS}, \mathsf{sid}, r)$ to this player.

Part II

Cryptographic Primitives

Chapter 3

Smooth Projective Hash Functions

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [CS02] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found numerous applications in various contexts in cryptography (for instance, [GL03, Kal05, BPV12], [**ACP09**]).

3.1 Definition and Classical Properties

Definition 1 (Smooth Projective Hashing System). *A Smooth Projective Hash Function over a language $\mathfrak{L} \subset \mathrm{X}$, is defined by five algorithms* $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$ *generates the global parameters* param *of the scheme, and the description of an $\mathcal{NP}$ language $\mathfrak{L}$;*
- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$*, outputs a hashing key* hk *for the language $\mathfrak{L}$;*
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W})$*, derives the projection key* hp*, using the hashing key* hk*;*
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W})$*, outputs a hash value v, thanks to the hashing key* hk*, and* W*;*
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}, w)$*, outputs the hash value v', thanks to* hp *and the witness w that $\mathrm{W} \in \mathfrak{L}$.*

In the following, we consider $\mathfrak{L}$ as a hard-partitioned subset of X, i.e. it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $\mathrm{X} \setminus \mathfrak{L}$.

The *correctness* of the SPHF assures that if $\mathrm{C} \in \mathrm{L}$ with w a witness of this membership, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, \mathrm{C}) = \mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, \mathrm{C}, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $\mathrm{C} \notin \mathrm{L}$, the hash value is *statistically* indistinguishable from a random element, even knowing hp.

A Smooth Projective Hash Function SPHF should satisfy the following properties:

- *Correctness*: Let $\mathrm{W} \in \mathfrak{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}, w)$.
- *Smoothness*: For all $\mathrm{W} \in \mathrm{X} \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, \mathrm{W}, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}) \in \mathbb{G} \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, \mathrm{W}, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}), v \xleftarrow{\$} \mathbb{G} \end{array} \right\}.$$

A third property called *Pseudo-Randomness*, is implied by the Smoothness on Hard Subset membership languages. If $\mathrm{W} \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time the following advantage is negligible

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, \mathrm{W}, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, \mathrm{W}, \mathsf{hp}, v) = 1]|$$

We define a third property in [**ABB⁺13**] (needed for our one-round PAKE protocol in combination with a commitment), called strong pseudo-randomness (see page 181). It is a strong version of the pseudo-randomness where the adversary is also given the hash value of a commitment of its choice (obviously not a simulated commitment, which can open to anything, though: See Section 4.3.1). This property only makes sense when the projection key does not depend on the word C to be hashed. It thus applies to KV-SPHF and CS-SPHF only (as defined in Section 3.2.1).

In [**BBC⁺13b**], we introduce a new notation for SPHF: for a language $\mathfrak{L}$, there exist a function Γ and a family of functions Θ, such that $\vec{u} \in \mathfrak{L}$, if and only if, $\Theta(\vec{u})$ is a linear combination $\vec{\lambda}$ of the rows of $\Gamma(\vec{u})$. We furthermore require that a user, who knows a witness of the membership $\vec{u} \in \mathfrak{L}$, can efficiently compute the linear combination $\vec{\lambda}$. The SPHF can now then be described as:

- $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key $\mathsf{hk} = \vec{\alpha}$ for the language $\mathfrak{L}$,
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \vec{u})$, derives the projection key $\mathsf{hp} = \vec{\gamma}(\vec{u})$,
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \vec{u})$, outputs a hash value $\mathrm{H} = \Theta(\vec{u}) \odot \vec{\alpha}$,
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), \vec{u}, \vec{\lambda})$, outputs the hash value $\mathrm{H}' = \vec{\lambda} \odot \vec{\gamma}(\vec{u})$.

We give in Section 3.4.2 an example of KV-SPHF for Cramer-Shoup encryption, both in classical and new notations.

3.2 Classification and Examples

3.2.1 Smoothness Adaptivity and Key Word-Dependence

The construction of one-round protocol exploits the very strong notion that we call KV-SPHF. Informally, while the GL-SPHF definition allows the projection key hp to depend on the word W, the KV-SPHF definition prevents the projection key hp from depending on W, as in the original CS-SPHF definition. In addition, the smoothness should hold even if W is chosen as an arbitrary function of hp. This models the fact the adversary can see hp before deciding which word W it is interested in. More formal definitions follow, where we denote Π the range of the hash function.

CS-SPHF

This is almost[1] the initial definition of SPHF, where the projection key hp does not depend on the word W (word-independent key), but the word W cannot be chosen after having seen hp for breaking the smoothness (non-adaptive smoothness). More formally, a CS-SPHF is ε-smooth if ProjKG does not use its input W and if, for any $\mathrm{W} \in \mathrm{X}\backslash\mathfrak{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, \mathrm{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}, \perp);\ \mathrm{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathfrak{L}, \mathrm{W})\}$$
$$\{(\mathsf{hp}, \mathrm{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}, \perp);\ \mathrm{H} \xleftarrow{\$} \Pi\}.$$

GL-SPHF

This is a relaxation, where the projection key hp can depend on the word W (word-dependent key). More formally, a GL-SPHF is ε-smooth if, for any $\mathrm{W} \in \mathrm{X}\backslash\mathfrak{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, \mathrm{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}, \mathrm{W});\ \mathrm{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathfrak{L}, \mathrm{W})\}$$
$$\{(\mathsf{hp}, \mathrm{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}, \mathrm{W});\ \mathrm{H} \xleftarrow{\$} \Pi\}.$$

KV-SPHF

This is the strongest SPHF, in which the projection key hp does not depend on the word W (word-independent key) and the smoothness holds even if W depends on hp (adaptive smoothness). More formally, a KV-SPHF is ε-smooth if ProjKG does not use its input W and, for any function f onto $\mathrm{X}\backslash\mathfrak{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, \mathrm{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}, \perp);\ \mathrm{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathfrak{L}, f(\mathsf{hp}))\}$$
$$\{(\mathsf{hp}, \mathrm{H}) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathfrak{L}, \perp);\ \mathrm{H} \xleftarrow{\$} \Pi\}.$$

[1] In the initial definition, the smoothness was defined for a word W randomly chosen from $\mathrm{X}\backslash\mathfrak{L}$, and not necessarily for any such word.

Remark. *One can see that a perfectly smooth (i.e. 0-smooth) CS-SPHF is also a perfectly smooth KV-SPHF, since each value* H *has exactly the same probability to appear, and so adaptively choosing* W *does not increase the above statistical distance. However, as soon as a weak word* W *can bias the distribution,* f *can exploit it.*

3.2.2 SPHF on Languages of Ciphertexts

We cover in [**BBC**$^+$**13a**] languages very general, but for the sake of clarity, and since the main applications need some particular cases only, we focus on SPHFs for languages of ciphertexts, whose corresponding plaintexts verify some relations. We denote these languages $\textsc{LofC}_{\mathsf{full\text{-}aux}}$.

The parameter full-aux will parse in two parts $(\mathsf{crs}, \mathsf{aux})$: the public part crs, known in advance, and the private part aux, possibly chosen later. More concretely, crs represents the public values: it will define the encryption scheme (and will thus contain the global parameters and the public key of the encryption scheme) with the global format of both the tuple to be encrypted and the relations it should satisfy, and possibly additional public coefficients; while aux represents the private values (indeed, unless specified differently, aux is assumed private): it will specify the relations, with more coefficients or constants that will remain private, and thus implicitly known by the sender and the receiver (as the expected password, for example, in PAKE protocols).

To keep aux secret, hp should not leak any information about it. We will thus restrict HashKG and ProjKG not to use the parameter aux, but just crs. This is a stronger restriction than required for our purpose, since one can use aux without leaking any information about it. But we already have quite efficient instantiations, and it makes everything much simpler to present.

3.2.3 SPHF on Cramer-Shoup Ciphertexts

Labeled Cramer-Shoup Encryption Scheme (CS)

We briefly review the CS labeled encryption scheme, where we combine all the public information in the encryption key. We thus have a group $\mathbb{G}$ of prime order p, with two independent generators $(g_1, g_2) \overset{\$}{\leftarrow} \mathbb{G}^2$, a hash function $\mathfrak{H}_{\mathrm{K}} \overset{\$}{\leftarrow} \mathcal{H}$ from a collision-resistant hash function family onto $\mathbb{Z}_p^*$, and a reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$. From 5 scalars $(x_1, x_2, y_1, y_2, z) \overset{\$}{\leftarrow} {\mathbb{Z}_p}^5$, one also sets $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. The encryption key is $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_{\mathrm{K}})$, while the decryption key is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z)$. For a message $m \in \{0,1\}^n$, with $\mathrm{M} = \mathcal{G}(m) \in \mathbb{G}$, the labeled Cramer-Shoup ciphertext is:

$$\mathrm{C} \overset{\mathsf{def}}{=} \mathsf{CS}(\ell, \mathsf{ek}, \mathrm{M}; r) \overset{\mathsf{def}}{=} (\boldsymbol{u} = (g_1^r, g_2^r), e = \mathrm{M} \cdot h^r, v = (cd^\xi)^r),$$

with $\xi = \mathfrak{H}_{\mathrm{K}}(\ell, \boldsymbol{u}, e) \in \mathbb{Z}_p^*$. If one wants to encrypt a vector of group elements $(\mathrm{M}_1, \ldots, \mathrm{M}_n)$, all at once in a non-malleable way, one computes all the individual ciphertexts with a common $\xi = \mathfrak{H}_{\mathrm{K}}(\ell, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_n, e_1, \ldots, e_n)$ for $v_1, \ldots, v_n$. Hence, everything done on tuples of ciphertexts will work on ciphertexts of vectors. In addition, the Cramer-Shoup labeled encryption scheme on vectors is IND-CCA under the DDH assumption.

The (known) GL-SPHF for CS

Gennaro and Lindell [GL03] proposed an SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta, \theta, \mu, \nu) \overset{\$}{\leftarrow} \mathbb{Z}_p^4$. The associated projection key, on a ciphertext $\mathrm{C} = (\boldsymbol{u} = (u_1, u_2) = (g_1^r, g_2^r), e = \mathcal{G}(m) \cdot h^r, v = (cd^\xi)^r)$, is $\mathsf{hp} = g_1^\eta g_2^\theta h^\mu (cd^\xi)^\nu \in \mathbb{G}$. Then, one can compute the hash value in two different ways, for the language $\textsc{LofC}_{\mathsf{ek},m}$ of the valid ciphertexts of $\mathrm{M} = \mathcal{G}(m)$, where $\mathsf{crs} = \mathsf{ek}$ is public but $\mathsf{aux} = m$ is kept secret:

$$\begin{aligned} \mathrm{H} \overset{\mathsf{def}}{=} \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), \mathrm{C}) &\overset{\mathsf{def}}{=} u_1^\eta u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu \\ &= \mathsf{hp}^r \overset{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), \mathrm{C}, r) \overset{\mathsf{def}}{=} \mathrm{H}'. \end{aligned}$$

A (new) KV-SPHF for CS

We give here the description of the first known KV-SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \overset{\$}{\leftarrow} \mathbb{Z}_p^5$; the associated projection key is the pair $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu}) \in \mathbb{G}^2$. Then one can compute the hash value in two different ways, for the language $\mathrm{LoFC}_{\mathsf{ek},m}$ of the valid ciphertexts of $\mathrm{M} = \mathcal{G}(m)$ under ek:

$$\begin{aligned} \mathrm{H} = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), \mathrm{C}) &\overset{\mathsf{def}}{=} u_1^{(\eta_1+\xi\eta_2)} u_2^{\theta} (e/\mathcal{G}(m))^{\mu} v^{\nu} \\ &= (\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r \overset{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), \mathrm{C}, r) = \mathrm{H}'. \end{aligned}$$

Theorem 1 ([**BBC⁺13b**]). *The above SPHF is a perfectly smooth (i.e. 0-smooth) KV-SPHF.*

The proof can be found in [**BBC⁺13b**] (page 162) as an illustration of our new framework.

3.3 Constructions

In all this work, we focus on the elliptic curve setting, which is currently the most studied. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^{\mathfrak{K}}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

In the protocols studied (PAKE and OT), we usually need to verify pairing product equations over bilinear groups (for instance to verify a signature), in which certain terms are private. In order to transmit them publicly, the prover starts with sending commitments on these values. In [GS08], Groth and Sahai proposed non-interactive zero-knowledge proofs of satisfiability of these equations by showing that the committed values actually satisfy the equation. The proofs consists of group elements, and is verified by a pairing equation derived from the statement. In our applications, the verification is indeed done implicitely, by the means of SPHFs: In case the "proof" is correct, then the hash value computed is correct. Otherwise, it is completely indistinguishable from a random value, thanks to the smoothness property.

In [**BBC⁺13a**], we show that all the proofs doable in the Groth-Sahai methodology can be done with GL-SPHFs, which can even handle unknown values in $\mathbb{G}_T$. In [**BBC⁺13b**], we present the first constructions of GL-SPHFs and KV-SPHFs able to handle multi-exponentiation equations without requiring pairings. This gives a full treatment for *linear* pairing product equations, which encompass Diffie-Hellman tuples, El Gamal encryptions, Cramer-Shoup encryptions, verification of BLS signatures... This allows for many cryptographic applications, such as one-round UC-secure PAKE, as we will see in Chapter 5. For *quadratic* pairing product equations (with a unknown variable on each side of the equation), however, we are able to give constructions for GL-SPHF, but KV-SPHF are still missing. This would have several applications, for instance for the verification of "strong" Boneh-Boyen signatures.

3.4 Additional Properties: Structure-Preserving SPHF

Similarly to structure-preserving signatures requiring the message, the signature and the public keys to be group elements, we propose in [**BC16**] the notion of structure-preserving Smooth Projective Hash Functions (SP-SPHF), where both words, witnesses and projection keys are group elements, and hash and projective hash computations are doable with simple pairing-product equations in the context of bilinear groups.

Since witnesses now become group elements, this allows a full compatibility with Groth and Sahai methodology [GS08], such that we can build, for instance, SPHFs that implicitly demonstrate the knowledge of a Groth-Sahai proof (the non-interactive zero-knowledge proof of knowledge now serving as a witness). This leads to interesting applications: As an example, we show in our paper that the UC-commitment from [FLM11] (while not fitting with the methodology of traditional SPHF from [**ABB⁺13**], see Section 4.3.3), is compatible with SP-SPHF and can be used to build UC protocols (see Sections 5.2.2 and 6.2.3).

We show in [**BC16**] how to transform every previously known pairing-less construction of SPHF to fit this methodology, and then propose several applications in which storing a group element as a witness allows to avoid the drastic restrictions that arise when building protocols secure against adaptive corruptions in the UC framework with a scalar as witness. Asking the witness to be a group element enables us to gain more freedom in the simulation (the discrete logarithm of this element and / or real extraction from a commitment). For instance, the simulator can always commit honestly to a random message, since it only needs to modify its witness in the equivocation phase. Furthermore, it allows to avoid bit-per-bit construction. Such design carries similarity with the publicly verifiable MACs from [KPW15], where the pairing operation allows to relax the verification procedure.

3.4.1 Definition

As we are in the context of structure-preserving cryptography, we assume the existence of a (prime order) bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e)$, and consider languages (sets of elements) $\mathfrak{L}$ defined over this group. The hash space is usually $\mathbb{G}_T$, the projection key space a group $\mathbb{G}_1^m \times \mathbb{G}_2^n$ and the witness space a group $\mathbb{G}_1^n \times \mathbb{G}_2^m$.

Definition 2 (Structure-Preserving Smooth Projective Hash Functions)**.** *A Structure-Preserving Smooth Projective Hash Function over a language $\mathfrak{L} \subset \mathrm{X}$ onto a set $\mathcal{H}$, is defined by 4 algorithms* (HashKG, ProjKG, Hash, ProjHash)*:*

- HashKG$(\mathfrak{L}, \mathsf{param})$*, outputs a hashing key* hk *for the language $\mathfrak{L}$;*
- ProjKG$(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W})$*, derives the projection key* hp *thanks to the hashing key* hk*.*
- Hash$(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \mathrm{W})$*, outputs a hash value $\mathrm{H} \in \mathcal{H}$, thanks to the hashing key* hk*, and* W
- ProjHash$(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), \mathrm{W}, w)$*, outputs the value $\mathrm{H}' \in \mathcal{H}$, thanks to* hp *and the witness w that $\mathrm{W} \in \mathfrak{L}$.*

We stress that, contrarily to classical SPHF, both hp, W and more importantly w are base group elements, and so live in the same space. Properties of correctness, smoothness and pseudo-randomness are then inherited by those of classical Smooth Projective Hash Functions.

3.4.2 Retro-compatibility

Constructing SP-SPHF is not that hard of a task. A first naive approach allows to transform every pairing-less SPHF into a SP-SPHF in a bilinear setting. It should be noted that while the resulting Hash/ProjHash values live in the target group, nearly all use cases encourage to use a proper hash function on them before computing anything using their value, hence the communication cost would remain the same. (Only applications where one of the party has to provide an additional proof that the ProjHash was honestly computed might be lost, but besides proof of negativity from **[BCV15]**, this never arises.)

To this goal, simply given a new generator $f \in \mathbb{G}_2$ and a scalar witness vector λ, one generates the new witness vector $\Lambda = [f \odot \vec{\lambda}]_2$. Words and projection keys belong to $\mathbb{G}_1$, and hash values to $\mathbb{G}_T$. Any SPHF can thus be transformed into an SP-SPHF in the way described in Figure 3.1.

Fig. 3.1 – Transformation of an SPHF into an SP-SPHF

	SPHF	SP-SPHF
Word $\vec{u}$	$[\vec{\lambda} \odot \Gamma(\vec{u})]_1$	$[\vec{\lambda} \odot \Gamma(\vec{u})]_1$
Witness w	$\vec{\lambda}$	$\vec{\Lambda} = [f \odot \vec{\lambda}]_2$
hk	$\vec{\alpha}$	$\vec{\alpha}$
hp $= [\vec{\gamma}(\vec{u})]_1$	$[\Gamma(\vec{u}) \odot \vec{\alpha}]_1$	$[\Gamma(\vec{u}) \odot \vec{\alpha}]_1$
Hash(hk, $\vec{u}$)	$[\Theta(\vec{u}) \odot \vec{\alpha}]_1$	$[f \odot \Theta(\vec{u}) \odot \vec{\alpha}]_{\mathrm{T}}$
ProjHash(hp, $\vec{u}$, w)	$[\vec{\lambda} \odot \vec{\gamma}(\vec{u})]_1$	$[\vec{\Lambda} \odot \vec{\gamma}(\vec{u})]_{\mathrm{T}}$

It should be noted that in case this does not weaken the subgroup decision assumption (k-MDDH in the following) linked to the original language, one can set $\mathbb{G}_1 = \mathbb{G}_2$.

We give in Figure 3.2 two examples of regular SPHFs on Diffie-Hellman and Cramer-Shoup encryption of M, where $\alpha = \mathcal{H}(\boldsymbol{u}, e)$, and their counterparts with SP-SPHF. ElGamal being a simplification of Cramer-Shoup, we skip the description of the associated SP-SPHF. We also give in Figure 3.3 the matricial version of Cramer-Shoup encryption, in which we denote by C' the Cramer-Shoup encryption C of M in which we removed M.

Fig. 3.2 – Example of conversion of classical SPHF into SP-SPHF (DH and CS)

	SPHF	SP-SPHF
DH	h^r, g^r	h^r, g^r
Witness w	r	g_2^r
hk	λ, μ	λ, μ
hp	$h^\lambda g^\mu$	$h^\lambda g^\mu$
$\mathsf{Hash}(\mathsf{hk}, \vec{u})$	$(h^r)^\lambda (g^r)^\mu$	$e((h^r)^\lambda (g^r)^\mu, g_2)$
$\mathsf{ProjHash}(\mathsf{hp}, \vec{u}, w)$	hp^r	$e(\mathsf{hp}, g_2^r)$
CS(M;r)	$h^r\mathrm{M}, f^r, g^r, (cd^\alpha)^r$	$h^r\mathrm{M}, f^r, g^r, (cd^\alpha)^r$
Witness w	r	g_2^r
hk	$\lambda_1, \lambda_2, \mu, \nu, \eta$	$\lambda_1, \lambda_2, \mu, \nu, \eta$
hp	$h^{\lambda_1} f^\mu g^\nu c^\eta, h^{\lambda_2} d^\nu$	$h^{\lambda_1} f^\mu g^\nu c^\eta, h^{\lambda_2} d^\nu$
$\mathsf{Hash}(\mathsf{hk}, \vec{u})$	$(h^r)^{\lambda_1+\alpha\lambda_2} (f^r)^\mu (g^r)^\nu ((cd^\alpha)^r)^\mu$	$e((h^r)^{\lambda_1+\alpha\lambda_2} (f^r)^\mu (g^r)^\nu ((cd^\alpha)^r)^\mu, g_2)$
$\mathsf{ProjHash}(\mathsf{hp}, \vec{u}, w)$	$(\mathsf{hp}_1 \mathsf{hp}_2^\alpha)^r$	$e(\mathsf{hp}_1 \mathsf{hp}_2^\alpha, g_2^r)$
(with $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$)		

Fig. 3.3 – Example of conversion of SPHF into SP-SPHF (CS in matricial notations)

	SPHF	SP-SPHF
CS(M;r)	$[hr + \mathrm{M}, \vec{\mathrm{A}}r, (c + d\alpha)r]$	$[hr + \mathrm{M}, \vec{\mathrm{A}}r, (c + d\alpha)r]_1$
$\vec{\mathrm{B}} : \begin{pmatrix} h \\ f \\ g \\ c \end{pmatrix}$	$\left[\vec{\mathrm{B}}r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ d \end{pmatrix} \alpha r + \begin{pmatrix} \mathrm{M} \\ 0 \\ 0 \\ 0 \end{pmatrix}\right]$	$\left[\vec{\mathrm{B}}r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ d \end{pmatrix} \alpha r + \begin{pmatrix} \mathrm{M} \\ 0 \\ 0 \\ 0 \end{pmatrix}\right]_1$
Witness w	r	$[r]_2$
hk	$\lambda_1, \lambda_2, \mu, \nu, \eta$	$\lambda_1, \lambda_2, \mu, \nu, \eta$
hp	$\left[\begin{pmatrix} \lambda_1 & \mu & \nu & \eta \end{pmatrix} \vec{\mathrm{B}}, \begin{pmatrix} \lambda_2 & 0 & 0 & \eta \end{pmatrix} \begin{pmatrix} h \\ 0 \\ 0 \\ d \end{pmatrix}\right]$	$[\mathsf{hp}]_1$
$\mathsf{Hash}(\mathsf{hk}, \vec{u})$	$\left[\begin{pmatrix} \lambda_1 + \alpha\lambda_2 & \mu & \nu & \eta \end{pmatrix} (\mathrm{C}')\right]$	$\left[\begin{pmatrix} \lambda_1 + \alpha\lambda_2 & \mu & \nu & \eta \end{pmatrix} (\mathrm{C}')\right]_{\mathrm{T}}$
$\mathsf{ProjHash}(\mathsf{hp}, \vec{u}, w)$	$[(\mathsf{hp}_1 + \alpha\mathsf{hp}_2)r]$	$[(\mathsf{hp}_1 + \alpha\mathsf{hp}_2)r]_{\mathrm{T}}$

3.4.3 Possible Applications

Nearly Constant 1-out-of-k Oblivious Transfer Using FLM. Recent pairing-based constructions ([CKWZ13] or our paper [**ABB$^+$13**]) of Oblivious Transfer use SPHF to mask each line of a database with the hash value of as SPHF on the language corresponding to the first flow being a commitment of the said line. But those constructions require special UC commitment on scalars (see Section 4.3.3), with equivocation and extraction capacities.

In 2011, [FLM11] proposed a UC commitment, whose decommitment operation is done via group elements. In section 6.2.3, we show how to combine the existing constructions with this efficient commitment using SP-SPHF, in order to obtain a very efficient round-optimal scheme where there is no longer a growing overhead due to the commitment.

Round-Optimal Password Authenticated Key Exchange with Adaptive Corruptions. Recent developments around SPHF-based PAKE have either lead to Round-Optimal PAKE in the BPR model [BPR00], or with static corruptions [KV11], [**BBC$^+$13b**]. In order to achieve round-optimality, we need in [**ABB$^+$13**] to do a bit-per-bit commitment of the password, inducing a communication cost proportional to the maximum password length.

In Section 5.2.2, we show how to take advantage of the SP-SPHF constructed on the FLM commitment [FLM11] to propose a One-Round PAKE UC secure against adaptive adversaries, providing a constant communication cost [**BC16**].

Using a ZKPK as a witness, Anonymous Credentials. Previous applications allow more efficient instantiations of protocols already using scalar-based SPHF. However, one can imagine additional scenarios, where a scalar-based approach may not be possible, due to the inherent nature of the witness used. For example, one could consider a strong authentication scenario, in which each user possesses an identifier delivered by an authority, and a certification on a commitment to this identifier, together with a proof of knowledge that this commitment is indeed a commitment to this identifier. (Such scenario can be transposed to the delivery of a Social Security Number, where a standalone SSN may not be that useful, but a SSN officially linked to someone is a sensitive information that should be hidden.) In this scenario, a user, who wants to access his record on a government service where he is already registered, should give the certificate, and then would use an implicit proof that this corresponds to his identifier. With our technique, the server would neither learn the certificate plaintext nor the user's identifier (if he did not possess it earlier), and the user would be able to authenticate only if his certificate is indeed on his committed identifier.

In our scenario, we could even add an additional step, such that Alice does not interact directly with Bob but can instead use a pawn named Carol. She could send to Carol a commitment to the signature on her identity, prove in a black box way that it is a valid signature on an identity, and let Carol do the interaction on her behalf. For example, to allow a medical practitioner to access some subpart of her medical record concerning on ongoing treatment, in this case, Carol would need to anonymously prove to the server that she is indeed a registered medical practitioner, and that Alice has given her access to her data.

Chapter 4

Commitments

4.1 Definition, Classical Properties and Ideal Functionality

We give here the informal security definitions for commitment schemes and refer the reader to [**ABB$^+$13**] for details. In a typical commitment scheme, there are two main phases. In a *commit* phase, the committer computes a commitment C for some message m and sends it to the receiver. Then, in an *opening* phase, the committer releases some information δ to the receiver which allows the latter to verify that C was indeed a commitment of m.

A commitment scheme $\mathcal{C}$ is thus defined by 3 algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, implicitly given as input to the other algorithms;
- $\mathsf{Commit}(m; r)$ produces a commitment C on the input message $m \in \mathcal{M}$, using the random coins $r \xleftarrow{\$} \mathcal{R}$, and also outputs the opening information w;
- $\mathsf{Decommit}(\mathrm{C}, m; \delta)$ decommits the commitment C using the opening information δ; it outputs the message m, or $\perp$ if the opening check fails.

To be useful in practice, a commitment scheme should satisfy two basic security properties. The first one is *hiding*, which informally guarantees that no information about x is leaked through the commitment C. The second one is *binding*, which guarantees that the committer cannot generate a commitment C that can be successfully opened to two different messages. These two properties can be obtained in a perfect, statistical or computational way, according to the power an adversary would need to break them. But essentially, a *perfectly binding* commitment scheme guarantees the uniqueness of the opening phase. This is achieved by an encryption scheme, which on the other hand provides the *computational hiding* property only, under the $\mathsf{IND\text{-}CPA}$ security. A *perfectly hiding* commitment scheme guarantees the perfect secrecy of m.

In addition to the binding and hiding properties, certain applications may require additional properties from a commitment scheme. One such property is *equivocability* [Bea96], for a *perfectly hiding* commitment scheme, which guarantees that a commitment C can be opened in more than a single way when in possession of a certain trapdoor information. The commitment admits an indistinguishable Setup phase that generates a trapdoor allowing to open a commitment in any way.

Another one is *extractability*, for a *perfectly binding* commitment scheme, which allows the computation of the message m committed in C when in possession of a certain trapdoor information. Again, an encryption scheme is an extractable commitment, where the decryption key is the trapdoor that allows extraction.

Yet another property that may also be useful for cryptographic applications is *non-malleability* [DDN00], which ensures that the receiver of a unopened commitment C for a message m cannot generate a commitment for a message that is related to m.

Though commitment schemes satisfying stronger properties such as *non-malleability*, *equivocability*, and *extractability* may be useful for solving specific problems, they usually stop short of guaranteeing security when composed with arbitrary protocols. To address this problem, Canetti and Fischlin [CF01] proposed an ideal functionality for commitment schemes in the universal composability (UC) framework

[Can01] which guarantees all these properties simultaneously and allow the schemes to remain secure even under concurrent compositions with arbitrary protocols. Unfortunately, they also showed that such commitment schemes can only be realized if one makes additional setup assumptions, such as the existence of a common reference string (CRS) [CF01], random oracles [HMQ04], or secure hardware tokens [Kat07].

A UC-secure commitment scheme provides all the properties previously given: it should be hiding and binding, but also extractable and equivocable, and even non-malleable. The ideal functionality is presented on Figure 4.1 and borrowed from [Can01, Lin11a]. A delayed output is an output first sent to the adversary $\mathcal{S}$ that eventually decides if and when the message is actually delivered to the recipient. This models denial of services from the adversary.

Fig. 4.1 – Ideal Functionality $\mathcal{F}_{\mathsf{mcom}}$ for Multiple Commitments

The functionality $\mathcal{F}_{\mathsf{mcom}}$ proceeds as follows, with session identifier sid and running with parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$, a security parameter $1^{\mathfrak{K}}$, and an adversary $\mathcal{S}$:

- Commit phase: Upon receiving a message of the form $(\mathsf{Commit}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, m)$ from P_i where $m \in \{0,1\}^{\mathsf{polylog}k}$, record the tuple $(\mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, m)$ and generate a delayed output $(\mathsf{receipt}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to P_j. Ignore further Commit-message with the same $(\mathsf{sid}, \mathsf{ssid})$.

- Reveal phase: Upon receiving a message $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid})$ from party P_i, if a tuple $(\mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, m)$ was previously recorded, then generate a delayed output $(\mathsf{reveal}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, m)$ to P_j. Ignore further reveal-message with the same $(\mathsf{sid}, \mathsf{ssid})$ from P_i.

4.2 Discussion and Correction of Lindell's Protocols

Several UC-secure commitment schemes in the CRS model have been proposed. Canetti and Fischlin [CF01] and Canetti, Lindell, Ostrovsky, and Sahai [CLOS02] proposed inefficient non-interactive schemes from general primitives. On the other hand, Damgård and Nielsen [DN02], and Camenish and Shoup [CS03] (among others) presented interactive constructions from several number-theoretic assumptions.

Lindell [Lin11a] presented the first very efficient commitment schemes proven in the UC framework. They can be viewed as combinations of Cramer-Shoup encryption schemes and Σ-protocols. He presented two versions, one proven against static adversaries (static corruptions), while the other can also handle adaptive corruptions. These two schemes have commitment lengths of only 4 and 6 group elements respectively, while their total communication complexity amounts to 14 and 19 group elements respectively. Their security relies on the classical Decisional Diffie-Hellman assumption in standard cryptographic groups. Fischlin, Libert and Manulis [FLM11] shortly after adapted the scheme secure against static corruptions by removing the interaction in the Σ-protocol using non-interactive Groth-Sahai proofs [GS08]. This transformation also makes the scheme secure against adaptive corruptions but at the cost of relying on the Decisional Linear assumption in symmetric bilinear groups. It thus requires the use of computationally expensive pairing computations for the receiver and can only be implemented over groupes twice[1] as large (rather than the ones that do not admit pairing computations).

We only give here the high-level ideas and refer the reader to [Lin11a], [**BCPV13**] for details and precise notations. The CRS consists of $(p, \mathbb{G}, g_1, g_2, c, d, h, h_1, h_2, \zeta, \mathfrak{H}_{\mathrm{K}})$, where $\mathbb{G}$ is a group of order p with generators g_1, g_2; $c, d, h \in \mathbb{G}$ are random elements in $\mathbb{G}$ and $h_1 = g_1{}^{\rho}$ and $h_2 = g_2{}^{\rho}$ for a random $\rho \in \mathbb{Z}_p$; $\mathfrak{H}_{\mathrm{K}}$ is randomly drawn from a collision-resistant hash function family $\mathcal{H}$. Intuitively $(p, \mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_{\mathrm{K}})$ is a Cramer-Shoup encryption key, $(p, \mathbb{G}, g_1, g_2, h_1, h_2)$ is the CRS of a dual-mode encryption scheme, and $(p, \mathbb{G}, g, \zeta) =$ is the CRS of a Pedersen commitment scheme. Finally, $\mathrm{G} : \{0,1\}^n \to \mathbb{G}$ is an efficiently computable and invertible mapping of a binary string to the group. The adaptive version of the protocol can be seen in Figure 4.3.

[1]It is possible to adapt the scheme from [FLM11] to asymmetric bilinear groups using the instantiation of Groth-Sahai proofs based on the Strong eXternal Diffie-Hellman assumption (as we did in [**BC16**]) but Lindell's scheme nevertheless remains more efficient.

4.2.1 Security Against Adaptive Corruptions

Lindell has proven both schemes secure under the DDH assumption, the former in details but a sketch of proof only for the latter. And actually, as noted by Lindell in the last version of [Lin11b], the security against adaptive corruptions might eventually not be guaranteed.

He indeed proves that no adversary can choose a message m' beforehand, and do a valid commit/decommit sequence to m' where the simulator extraction, at the end of the commit phase, would output an m different from m'. However this is not enough as an adversary could still do a valid commit/decommit sequence to m' where the simulator extraction at the end of the commit phase would output an m different from m'. The difference between the two experiments is how much the adversary controls the value m': in the former m' has to be chosen beforehand, while in the latter m' is any value different from m.

We describe in [**BCPV13**] such a situation in which the adversary $\mathcal{A}$ plays as P_i, and makes the simulator extract the value m, while in fact committing (or actually opening) to another value m'.

Any extraction done on C at the end of the commit phase would lead the simulator to believe to a commit to m, however the valid decommit outputs m'. Note however that this attack does not succeed very often since one needs to invert G on a random value, so that the preimage exists and can be parsed as $(m', \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$.

We stress that this possible inconsistency comes from the move forward of the proof in the commit phase, even before the message m is strongly committed. The first protocol does not suffer from this issue.

4.2.2 A Simple Patch

In order to avoid the above concern, a simple patch consists in committing $x = \mathrm{G}(m, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ in the second Pedersen commitment c_p^2. This leads to the simple change in the protocol presented on Figure 4.2, where m is now strongly committed before the proof, and then the previous issue does not occur anymore.

Fig. 4.2 – Simple Patch to the Adaptive Version of Lindell's Protocol

The commit phase

5. P_i picks $s, k_2 \xleftarrow{\$} \mathbb{Z}_p$ and computes $(\alpha, \beta, \gamma, \delta) = ({g_1}^s, {g_2}^s, h^s, (cd^\omega)^s)$.
   He then computes and sends $c_p^2 = \mathsf{Ped}(\underline{x}, \mathfrak{H}_{\mathrm{K}}(\alpha, \beta, \gamma, \delta); k_2)$ to P_j.

4.2.3 Our Optimization of the Commitments Protocols

We kept in [**BCPV13**] (and in Figures 4.2 and 4.3 here) the original notations for describing Lindell's schemes and the patch we proposed, but as we did earlier in [**BBC$^+$13a**], we can note that C is actually a Cramer-Shoup encryption of $x = \mathrm{G}(m, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$, and $(\alpha, \beta, \gamma, \delta)$ is a partial Cramer-Shoup encryption of 1 with the same ω as in the first ciphertext: the double Cramer-Shoup encryption of (x, x') is denoted by $\mathsf{DCS}(x, x'; r, s) = (\mathrm{C}_1, \mathrm{C}_2)$, where

- C_1 is a real Cramer-Shoup encryption $\mathrm{C}_1 = \mathsf{CS}(x; r)$ of x for a random $r \xleftarrow{\$} \mathbb{Z}_p$: $\mathrm{C}_1 = (\boldsymbol{u}_1 = ({g_1}^r, {g_2}^r), e_1 = x \cdot h^r, v_1 = (cd^\omega)^r)$, where v_1 is computed afterwards with $\omega = \mathfrak{H}_{\mathrm{K}}(\boldsymbol{u}_1, e_1)$;
- C_2 is a partial Cramer-Shoup encryption $\mathrm{C}_2 = \mathsf{PCS}(x'; \omega, s)$ of x' for a random $s \xleftarrow{\$} \mathbb{Z}_p$ with the above ω value: $\mathrm{C}_2 = (\boldsymbol{u}_2 = ({g_1}^s, {g_2}^s), e_2 = x' \cdot h^s, v_2 = (cd^\omega)^s)$, where v_2 is computed directly with the above $\omega = \mathfrak{H}_{\mathrm{K}}(\boldsymbol{u}_1, e_1)$.

In addition, when ω is fixed, we have an homomorphic property: if $(\mathrm{C}_1, \mathrm{C}_2) = \mathsf{DCS}(x, x'; r, s)$, with a common ω, the component-wise product $\mathrm{C}_1 \times \mathrm{C}_2 = \mathsf{PCS}(x \times x'; \omega, r + s)$. In particular, we can see the last tuple $(\alpha u_1^\varepsilon, \beta u_2^\varepsilon, \gamma e^\varepsilon, \delta v^\varepsilon)$ as $\mathrm{C}_2 \times \mathrm{C}_1^\varepsilon$. It should thus be $\mathsf{PCS}(x^\varepsilon; \omega, \varepsilon r + s) = \mathsf{PCS}(x^\varepsilon; \omega, z)$, which is the final check. We now use these new notations (see [**BCPV13**] for their specific definitions and for details on the protocols) in the following.

Improvement of the Static Protocol

The improvement presented in [**BCPV13**] consists in noting that the receiver can directly send the value ε in the decommit phase, without having to send a commitment first. To allow this, we simply ask the sender to send a Pedersen commitment of $\mathrm{C}_2 = (\alpha, \beta, \gamma, \delta)$ prior to receiving ε. This reduces the number of flows of the decommit phase (from 5 downto 3) and the number of elements sent by the receiver (from 2 group elements and 3 scalars down to only 1 scalar, the challenge), simply increasing the number of elements sent by the sender by 1 group element and 1 scalar (the Pedersen commitment).

Improvement of the Adaptive Protocol

As for the static version of the protocol, the main improvement consists in noting that the receiver can directly send the value ε, without having to send an encryption before. To allow this, we simply ask the sender to send his two Pedersen commitments prior to receiving ε.

This reduces, in the commit phase, the number of rounds (from 5 downto 3) and the number of elements sent by the receiver (from 2 group elements and 3 scalars down to only 1 scalar, the challenge). Contrary to the static version, there is no additional cost. This is illustrated in Figure 4.3, which sums up the differences between Lindell's protocol and ours, in the same setting: UC-security against adaptive corruption with erasures.

In addition, in order to slightly increase the message space from $n - \log_2(n)$ to n, we move the sensitive prefix $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ into the second Pedersen.

Eventually, in order to definitely exclude the security concerns presented above, we include the value x to the second Pedersen to prevent the adversary from trying to open his commitment to another value.

4.3 Additional Properties and New Constructions

4.3.1 Extractable and Equivocable Commitment Schemes

Following the work of Canetti and Fischlin [CF01], we aimed in [**ABB$^+$13**] to build *non-interactive* commitment schemes which can simultaneously guarantee *non-malleability*, *equivocability* and *extractability* properties. To this end, we first define a new notion of commitment scheme, called E^2-commitments (page 176), for which there exists an alternative setup algorithm, whose output is computationally indistinguishable from that of a normal setup algorithm and which outputs a common trapdoor that allows for both equivocability and extractability: this trapdoor not only allows for the extraction of a committed message, but it can also be used to create simulated commitments which can be opened to any message.

To define the security of E^2-schemes, we first extend the security notions of standard equivocable commitments and extractable commitments to the E^2-commitment setting: Since the use of a common trapdoor for equivocability and extractability could potentially be exploited by an adversary to break the extractability or equivocability properties of an E^2-commitment scheme, we define stronger versions of these notions, which account for the fact that the same trapdoor is used for both extractability or equivocability. In particular, in these stronger notions, the adversary is given oracle access to the simulated commitment and extractor algorithms.

Finally, after defining the security of E^2-schemes, we further show that these schemes remain secure even under arbitrary composition with other cryptographic protocols. More precisely, we show that any labeled E^2–commitment scheme which meets the strong versions of the equivocability or extraction notions is a non-interactive UC-secure (multiple) commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and a single global CRS (page 177).

4.3.2 Constructions

After defining in [**ABB$^+$13**] the notion of E^2-commitment, we give an instantiation of such a scheme (see page 178), inspired by the commitment schemes in [CF01, CLOS02], [**ACP09**]. Like the construction in [**ACP09**], it combines a variant of the Cramer-Shoup encryption scheme (as an extractable commitment scheme) and an equivocable commitment scheme in order to be able to simultaneously achieve both equivocability and extractability. However, unlike the construction in [**ACP09**], we rely on Haralambiev's perfectly hiding commitment [Har11, Section 4.1.4], instead of the Pedersen commitment [Ped92].

Fig. 4.3 – Comparison of Lindell's Adaptive Protocol (above) and Ours (below)

The commit phase

$$\begin{array}{rcl}
x = \mathrm{G}(m, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j) & & \\
r \overset{\$}{\leftarrow} \mathbb{Z}_p, k_1 \overset{\$}{\leftarrow} \mathbb{Z}_p & & \\
\mathrm{C} = \mathsf{CS}(x; r) & & \\
c_p^1 = \mathsf{Ped}(\mathfrak{H}_\mathrm{K}(\mathrm{C}); k_1) & \xrightarrow{c_p^1} & \mathrm{R}, \mathrm{S} \overset{\$}{\leftarrow} \mathbb{Z}_p, \varepsilon \overset{\$}{\leftarrow} \{0,1\}^n \\
s \overset{\$}{\leftarrow} \mathbb{Z}_p, k_2 \overset{\$}{\leftarrow} \mathbb{Z}_p & \xleftarrow{c'} & c' = ({g_1}^\mathrm{R} {g_2}^\mathrm{S}, {h_1}^\mathrm{R} {h_2}^\mathrm{S} \mathrm{G}(\varepsilon)) \\
(\alpha, \beta, \gamma, \delta) = ({g_1}^s, {g_2}^s, h^s, (cd^\omega)^s) & & \\
c_p^2 = \mathsf{Ped}(\mathfrak{H}_\mathrm{K}(\alpha, \beta, \gamma, \delta)); k_2) & \xrightarrow{c_p^2} & \\
\text{Aborts if } c' \text{ inconsistent} & \xleftarrow{\mathrm{R}, \mathrm{S}, \varepsilon} & \\
z = s + \varepsilon r, \text{ erases } r, s & \xrightarrow{k_1, \mathrm{C}} & \text{Aborts if } c_p^1 \text{ inconsistent}
\end{array}$$

The decommit phase

$$\begin{array}{cl}
\xrightarrow{(m, \alpha, \beta, \gamma, \delta, k_2, z)} & x = \mathrm{G}(m, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j) \\
& \text{checks } c_p^2 \text{ and whether} \\
& g_1^z = \alpha {u_1}^\varepsilon, {g_2}^z = \beta {u_2}^\varepsilon \\
& h^z = \gamma (e/x)^\varepsilon, (cd^\omega)^z = \delta v^\varepsilon
\end{array}$$

The commit phase

$$\begin{array}{rcl}
x = \mathrm{G}(m) & & \\
r \overset{\$}{\leftarrow} \mathbb{Z}_p, s \overset{\$}{\leftarrow} \mathbb{Z}_p & & \\
(\mathrm{C}_1, \mathrm{C}_2) = \mathsf{DCS}(x; 1; r, s) & & \\
k_1, k_2 \overset{\$}{\leftarrow} \mathbb{Z}_p & & \\
c_p^1 = \mathsf{Ped}(\mathfrak{H}_\mathrm{K}(\mathrm{C}_1); k_1) & & \\
c_p^2 = \mathsf{Ped}(\mathfrak{H}_\mathrm{K}(\mathrm{C}_2, x, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j); k_2) & \xrightarrow{c_p^1, c_p^2} & \\
& \xleftarrow{\varepsilon} & \varepsilon \overset{\$}{\leftarrow} \mathbb{Z}_p \\
z = s + \varepsilon r, \text{ erases } r, s & \xrightarrow{k_1, \mathrm{C}_1} & \text{Aborts if } c_p^1 \\
& & \text{inconsistent}
\end{array}$$

The decommit phase

$$\begin{array}{cl}
\xrightarrow{(m, \mathrm{C}_2, k_2, z)} & x = \mathrm{G}(m) \\
& \text{checks } c_p^2 \text{ and whether} \\
& g_1^z = \alpha {u_1}^\varepsilon, {g_2}^z = \beta {u_2}^\varepsilon \\
& h^z = \gamma (e/x)^\varepsilon, (cd^\omega)^z = \delta v^\varepsilon
\end{array}$$

Since the opening value of Haralambiev's scheme is a group element that can be encrypted in one ElGamal-like ciphertext to allow extractability, this globally leads to a better communication and computational complexity for the commitment. The communication is linear in $m \cdot \mathfrak{K}$, where m is the bit-length of the committed value and $\mathfrak{K}$ the security parameter. This is significantly better than the extractable commitment construction in [**ACP09**] which was linear in $m \cdot \mathfrak{K}^2$, but asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m.

4.3.3 SPHF-Friendly Commitments

Definition

Our goal in [**ABB⁺13**] was to build non-interactive E^2-commitments, to which smooth projective hash functions could be efficiently associated. Unfortunately, achieving this goal is not so easy due to the equivocability property of E^2-commitments. To understand why, let X be the domain of an SPHF function and let L be some underlying NP language such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. As seen in Chapter 3, a key property of these SPHF functions that makes them so useful for applications such as PAKE and OT is that, for words C in L, their values can be computed using either a *secret* hashing key hk or a *public* projected key hp together a witness w to the fact that C is indeed in L. A typical example of a language in which we are interested is the language L_x corresponding to the set of elements $\{C\}$ such that C is a valid commitment of x. Unfortunately, when commitments are equivocable, the language L_x containing the set of valid commitments of x may not be well defined since a commitment C could potentially be opened to any x. To get around this problem and be able to use SPHF with E^2-commitments, we show in [**ABB$^+$13**] that it suffices for an E^2-commitment scheme to satisfy two properties (see pages 179 and 176). The first one is the stronger version of the equivocability notion, which guarantees that equivocable commitments are computationally indistinguishable from normal commitments, even when given oracle access to the simulated commitment and extractor algorithms. The second one, which is called *robustness*, is new and guarantees that commitments generated by polynomially-bounded adversaries are perfectly binding. Finally, we say that a commitment scheme is *SPHF-friendly* if it satisfies both properties and if it admits an SPHF on the languages L_x.

Constructions

We show in [**ABB$^+$13**] that the E^2-commitment scheme proposed in the paper is also SPHF-friendly (see page 181). We generalize it in [**BC15**] by giving a generic construction of SPHF-friendly commitments from two simple blocks: a collision-resistant chameleon hash (CH) function which is verifiable (either publicly or for the receiver only) and an SPHF-friendly CCA encryption scheme. The extra requirement on the CH function is simple to achieve as soon as only classical algebraic operations are applied to the randomness, and SPHF-friendly encryption is now well-known since [CS02], with several instances (contrary to SPHF-friendly commitments, which is a difficult task). We give in particular an instanciation based on DDH.

The Case of FLM's Commitment

As already said in Section 4.3.2, the complexity of the commitment given in [**ABB$^+$13**] is asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m, but the latter proposals are not SPHF-friendly since they are not robust.

Thus, the UC-commitment from [FLM11] do not fit the methodology of traditional SPHF from [**ABB$^+$13**] and cannot be used as is in the PAKE and OT protocols proposed in [**ABB$^+$13**]. However, we show in [**BC16**] that it is indeed compatible with our new notion of SP-SPHF and can be used to build UC protocols (see pages 262, 264 and 265). As a side contribution, we first generalize in this paper this commitment from DLin to the $k-\mathsf{MDDH}$ assumption from [EHK$^+$13] (see page 261).

Part III

Cryptographic Protocols

Chapter 5

Password-Authenticated Key-Exchange

5.1 Security Definition in the UC Framework

The PAKE ideal functionality $\mathcal{F}_{pw\mathrm{KE}}$, presented on Figure 5.1, was described in [CHK$^+$05]. The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as compromised), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of wrong guess (the session is then marked as interrupted), the two players are given independently-chosen random keys. A session that is nor compromised nor interrupted is called fresh, which is its initial status.

Choosing the Passwords. First note that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the passwords are related passwords, used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

Defining a New Session. In the initialization step, the functionality merely waits for the players to notify their interest in participating to the protocol. More precisely, we assume that each player starts a new session with the query (NewSession, sid, P_i, P_j, pw_i, role), where P_i is the player's identity, P_j that of the player with which it would like to establish a communication, pw_i its password and role its role (client or server). The player's session is then denoted as fresh.

Modeling Dictionary Attacks. The adversary is awarded the right to try to guess the password of a player via TestPwd queries (this models the passwords' vulnerability as in an online attack in the real world: If the adversary correctly guessed the password, it succeeded in its impersonation attempt). Each player's session is initially fresh. A correct guess makes it become compromised, whereas an incorrect guess makes it become interrupted. Once the key is established, all the records are completed. Changing the fresh status of a record when a password guess (correct or not) happened or when a (valid) key was established enables to limit the number of guesses to at most one per player.

Generating the Session Key. The generation is initiated by the adversary, which enables it to decide at which exact moment the key has to be sent to the players. In particular, it can choose the exact moment where it can try to corrupt or impersonate a player (recall that $\mathcal{S}$ can only try to guess a password when the session of the player is *fresh*, which is not the case anymore after a NewKey query). If the two players own the same password and their sessions are fresh, they receive the same, uniformly distributed, key. If one of their sessions is interrupted, they obtain independent, randomly chosen keys. Otherwise, if one of the players is corrupted or its session compromised, the adversary is granted the right to choose its key.

Corruptions. In case of corruption, the adversary learns the password of the corrupted player, and after the NewKey-query, it additionally learns the session key.

Fig. 5.1 – Ideal Functionality for PAKE $\mathcal{F}_{\mathrm{pwKE}}$

The functionality $\mathcal{F}_{\mathrm{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathrm{P}_1, \ldots, \mathrm{P}_n$ via the following queries:

- **Upon receiving a query $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pw})$ from party P_i:**

  Send $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to $\mathcal{S}$. If this is the first NewSession query, or if this is the second NewSession query and there is a record $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_j, \mathrm{P}_i, \mathsf{pw}')$, then record $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pw})$ and mark this record fresh.

- **Upon receiving a query $(\mathsf{TestPwd}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathsf{pw}')$ from the adversary S:**

  If there is a record of the form $(\mathrm{P}_i, \mathrm{P}_j, \mathsf{pw})$ which is fresh, then do: If $pw = pw'$, mark the record compromised and reply to $\mathcal{S}$ with "correct guess". If $\mathsf{pw} \neq \mathsf{pw}'$, mark the record interrupted and reply with "wrong guess".

- **Upon receiving a query $(\mathsf{NewKey}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathsf{sk})$ from the adversary S:**

  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pw})$, and this is the first NewKey query for P_i, then:

  – If this record is compromised, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk})$ to player P_i.
  – If this record is fresh, and there is a record $(\mathrm{P}_j, \mathrm{P}_i, \mathsf{pw}')$ with $\mathsf{pw}' = \mathsf{pw}$, and a key sk' was sent to P_j, and $(\mathrm{P}_j, \mathrm{P}_i, \mathsf{pw})$ was fresh at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk}')$ to P_i.
  – In any other case, pick a new random key sk' of length $\mathfrak{K}$ and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pw})$ as completed.

5.2 Constructions of One-Round PAKE

5.2.1 Katz and Vaikuntanathan Smooth Projective Hash Functions

Katz and Vaikuntanathan [KV11] were the first to propose a *practical* one-round PAKE, where the two players just have to send simultaneous flows to each other, that depend on their own passwords only. More precisely, each flow just consists of an IND-CCA ciphertext of the password and an SPHF projection key for the correctness of the partner's ciphertext (the word is the ciphertext and the witness consists of the random coins of the encryption). The shared secret key is eventually the product of the two hash values, as in the KOY and GL protocols.

Because of the simultaneous flows, one flow cannot explicitly depend on the partner's flow, which makes impossible the use of the Gennaro and Lindell SPHF (named GL-SPHF, see Section 3.2.1), in which the projection key depends on the word (the ciphertext here). On the other hand, the adversary can wait for the player to send his flow first, and then adapt its message, which requires stronger security notions than the initial Cramer and Shoup SPHF (named CS-SPHF), in which the smoothness does not hold anymore if the word is generated after having seen the projection key. This led Katz and Vaikuntanathan to provide a new definition for SPHF (named KV-SPHF), where the projection key depends on the hashing key only, and the smoothness holds even if the word is chosen after having seen the projection key.

They also proposed another construction of one-round PAKE, provably secure against static corruptions in the UC framework. To achieve such a level of security, the simulator has to be more powerful: it should be able to make a successful execution after a dummy simulation, with a wrong password. To this aim, Katz and Vaikuntanathan allowed the simulator to extract the hashing key of the SPHF, to allow it to compute afterwards the hash value on any word, even outside the language. More precisely, each player additionally encrypts his hashing key to allow the key recovery by the simulator, so that the latter can compute the hash value even when a dummy password has initially been committed, whereas a success is expected.

However, previous SPHF known on Cramer-Shoup ciphertexts were GL-SPHF only. For their one-round PAKE, Katz and Vaikuntanathan did not manage to construct such a KV-SPHF for an efficient IND-CCA encryption scheme. They then suggested to use the Naor and Yung approach [NY90], with an ElGamal-like encryption scheme and a *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proof [Sah99]. Such an SS-NIZK proof is quite costly in general. They suggested to use Groth-Sahai [GS08] proofs in bilinear groups and the linear encryption [BBS04] which leads to a PAKE secure under the DLin assumption with a ciphertext consisting of 66 group elements and a projection key consisting of 4 group elements. As a consequence, the two players have to send 70 group elements each, which is far more costly than the KOY protocol, but it is one-round only.

More recent results on SS-NIZK proofs or IND-CCA encryption schemes, in the discrete logarithm setting, improved on that: Libert and Yung [LY12] proposed a more efficient SS-NIZK proof of plaintext equality in the Naor-Yung-type cryptosystem with ElGamal-like encryption. The proof can be reduced from 60 to 22 group elements and the communication complexity of the resulting PAKE is decreased to 32 group elements per user. Jutla and Roy [JR12] proposed relatively-sound NIZK proofs as an efficient alternative to SS-NIZK proofs to build new publicly-verifiable IND-CCA encryption schemes. They can then decrease the PAKE communication complexity to 20 group elements per user. In any case, one can remark that all one-round PAKE schemes require pairing computations.

5.2.2 Constructions of One-Round PAKE in the UC Framework

In [**BBC$^+$13b**], our first contribution is the description of an instantiation of KV-SPHF on Cramer-Shoup ciphertexts, and thus the first KV-SPHF on an efficient IND-CCA encryption scheme (see page 143). We thereafter use it within the above KV framework for one-round PAKE [KV11], in the BPR security model. Our scheme (described page 143) just consists of 6 group elements in each direction under the DDH assumption (4 for the ciphertext, and 2 for the projection key). This has to be compared with the 20 group elements, or more, in the best constructions discussed above, which all need pairing-friendly groups and pairing computations, or with the KOY protocol that has a similar complexity but with three sequential flows.

In [**ABB$^+$13**], we provide a generic construction of a one-round UC-secure PAKE from any SPHF-friendly commitment, verifying an additional property called strong pseudo-randomness (page 184). The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS. In addition to being the first one-round adaptively secure PAKE, our new scheme also enjoys a much better communication complexity than previous adaptively secure PAKE schemes. For instance, in comparison to the PAKE in [**ACP09**], which was in 2013 the most efficient adaptively secure PAKE, the new scheme gains a factor of $\mathfrak{K}$ in the overall communication complexity, where $\mathfrak{K}$ is the security parameter. However, unlike this scheme, the new construction requires pairing-friendly groups.

These constructions follow the Gennaro-Lindell methodology with variation of the Canetti-Fischlin commitment [CF01]. However their communication size is growing in the size of the passwords, which is leaking information about an upper-bound on the password used in each exchange. Besides being an efficiency problem, it is over all a security issue in the UC framework. Indeed, the simulator somehow has to "guess" the length of the password of the player it simulates, otherwise it is unable to equivocate the commitment (since the commitment reveals the length of the password it commits to). Since such a guess is impossible, the apparently only solution to get rid of this limitation seems to give the users an upper-bound on the length of their passwords and to ask them to compute commitments of this length, which leads to costly computations.

In [**BC16**], we provide an instantiation of a one-round UC-secure PAKE which does not leak any information about the length of the password used, under any $k-$ MDDH assumption, combining the UC commitment from [FLM11] (generalized to the $k-$ MDDH assumption from [EHK$^+$13]) and the associated SP-SPHF that we propose (page 264). We show that the UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS, and that our scheme enjoys a much better communication complexity (see Figure 5.2 for a comparison, in particular for the SXDH version[1]). Only

[1]We omit our paper [**BC15**] from this table, as its contribution is to widen the construction to non-pairing based hypotheses.

[JR14] achieves a slightly better complexity as ours, but only for SXDH, while ours easily extends to $k-$MDDH. Furthermore, our construction is an extension to SP-SPHF of well-known classical constructions based on SPHF, which makes it simpler to understand.

Fig. 5.2 – Comparison with existing UC-secure PAKE schemes where |password| = m

	Adaptive	One-round	Communication complexity	Assumption
[**ACP09**]	yes	no	$2 \times (2m + 22m\mathfrak{K}) \times \mathbb{G}$ + OTS	DDH
[KV11]	no	yes	$\approx 2 \times 70 \times \mathbb{G}$	DLIN
[**BBC$^+$13b**]	no	yes	$2 \times 6 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH
[**ABB$^+$13**]	yes	yes	$2 \times 10m \times \mathbb{G}_1 + 2 \times m \times \mathbb{G}_2$	SXDH
[JR14]	yes	yes	$4 \times \mathbb{G}_1 + 4 \times \mathbb{G}_2$	SXDH
[**BC16**]	yes	yes	$2 \times (k+3) \times \mathbb{G}_1$	k-MDDH
			$+2 \times (k+3+k(k+1)) \times \mathbb{G}_2$	
[**BC16**]	yes	yes	$2 \times 4 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH

We present in Figure 5.3 the PAKE protocol from [**BC16**], which is constant-size, round-optimal and UC-secure against adaptive corruptions (see the paper for details and notations, in particular the description of FLM's commitment, page 260). It builds upon the protocol proposed in [**ABB$^+$13**], using the SP-SPHF technique (see Section 3.4) to avoid the apparent impossibility to use FLM's commitment.

The language $\mathcal{L}_{\mathsf{pw}_i}$ is then the language of valid Cramer-Shoup encryptions of the embedded password $\mathrm{G}(\mathsf{pw}_i)$, consistent with the randomness committed in the second part, and the rest of the label.

Theorem 2 ([**BC16**]). *The Password Authenticated Key Exchange scheme described in Figure 5.3 is UC-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

Fig. 5.3 – One-Round UC-Secure PAKE from the revisited FLM's Commitment

CRS: $\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}})$.

Protocol execution by P_i **with** pw_i**:**

1. P_i generates $\mathsf{hk}_i \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_{\mathsf{pw}_i})$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathcal{L}_{\mathsf{pw}_i})$ and erases any random coins used for the generation
2. P_i computes $([\vec{\mathrm{C}}_i]_1, [\vec{\mathrm{R}}_i]_2, [\vec{\Pi}_i]_1) = \mathsf{Com}^{\ell_i}(\mathsf{crs}, \mathsf{pw}_i, \mathsf{sid}, \mathsf{cid}, \mathrm{P}_i, \mathrm{P}_j)$ with label $\ell_i = (\mathsf{sid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{hp}_i)$
3. P_i stores $[\vec{\Pi}_i]_1$, completely erases random coins used by Com and sends $\mathsf{hp}_i, [\vec{\mathrm{C}}_i]_1, [\vec{\mathrm{R}}_i]_2$ to P_j

Key computation: Upon receiving $\mathsf{hp}_j, [\vec{\mathrm{C}}_j]_1, [\vec{\mathrm{R}}_j]_2$ from P_j

1. P_i computes $\mathrm{H}'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathcal{L}_{\mathsf{pw}_i}, \ell_i, [\vec{\mathrm{C}}_i]_1, [\vec{\mathrm{R}}_i]_2), [\Pi_i]_1))$ and $\mathrm{H}_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathcal{L}_{\mathsf{pw}_i}, \ell_j, [\vec{\mathrm{C}}_j]_1, [\vec{\mathrm{R}}_j]_2))$ with $\ell_j = (\mathsf{sid}, \mathrm{P}_j, \mathrm{P}_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = \mathrm{H}'_i \cdot \mathrm{H}_j$ and erases everything else, except pw_i.

5.3 Extension to Language-Authenticated Key Exchange

5.3.1 Definition and Ideal Functionality

Password-Authenticated Key Exchange schemes presented above allow users to generate a strong cryptographic key based on a shared "human-memorable" (*i.e.* low-entropy) password without requiring a public-key infrastructure. In this setting, an adversary controlling all communication in the network should not be able to mount an off-line dictionary attack.

The concept of *Secret Handshakes* has been introduced in 2003 by Balfanz, Durfee, Shankar, Smetters, Staddon and Wong [BDS$^+$03] (see also [JL09b, AKB07]). It allows two members of the same group to identify each other secretly, in the sense that each party reveals his affiliation to the other only if they are members of the same group. At the end of the protocol, the parties can set up an ephemeral session key for securing further communication between them and an outsider is unable to determine if the handshake succeeded. In case of failure, the players do not learn any information about the other party's affiliation.

Credential-Authenticated Key Exchange (CAKE) was presented in 2010 by Camenisch, Casati, Groß and Shoup [CCGS10]. In this primitive, a common key is established if and only if a specific relation is satisfied between credentials hold by the two players. This primitive includes variants of PAKE and Secret Handshakes, and namely Verifier-based PAKE, where the client owns a password pw and the server knows a one-way transformation v of the password only. It prevents massive password recovering in case of server corruption. The two players eventually agree on a common high entropy secret if and only if pw and v match together, and off-line dictionary attacks are prevented for third-party players.

We propose in [**BBC$^+$13a**] a new primitive that encompasses most of the previous notions of authenticated key exchange. It is closely related to CAKE and we call it LAKE, for *Language-Authenticated Key-Exchange*, since parties establish a common key if and only if they hold credentials that belong to specific (and possibly independent) languages. The definition of the primitive is more practice-oriented than the definition of CAKE from [CCGS10] but the two notions are very similar. In particular, the new primitive enables privacy-preserving authentication and key exchange protocols by allowing two members of the same group to secretly and privately authenticate to each other without revealing this group beforehand.

In order to define the security of this primitive, we use the UC framework and an appropriate definition for languages that permits to dissociate the public part of the policy, the private common information the users want to check and the (possibly independent) secret values each user owns that assess the membership to the languages. We provide an ideal functionality for LAKE and give efficient realizations of the new primitive (for a large family of languages) secure under classical mild assumptions, in the standard model (with a common reference string – CRS), with static corruptions.

We significantly improve the efficiency of several CAKE protocols [CCGS10] for specific languages and we enlarge the set of languages for which we can construct practical schemes. Notably, we obtain a very practical realization of Secret Handshakes and a Verifier-based Password-Authenticated Key Exchange.

Language Definition

We define in [**BBC$^+$13a**] a simpler formalism for languages to be considered for SPHF than the one we formalized in [**ACP09**]. It is nevertheless more general: We consider any efficiently computable binary relation $\mathcal{R} : \{0,1\}^* \times \mathcal{P} \times \mathcal{S} \to \{0,1\}$, where the additional parameters $\mathsf{pub} \in \{0,1\}^*$ and $\mathsf{priv} \in \mathcal{P}$ define a language $\mathrm{L}_{\mathcal{R}}(\mathsf{pub}, \mathsf{priv}) \subseteq \mathcal{S}$ of the words W such that $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, \mathrm{W}) = 1$:

- pub are public parameters;
- priv are private parameters the two players have in mind, and they should think to the same values: they will be committed to, but never revealed;
- W is the word the sender claims to know in the language: it will be committed to, but never revealed.

Our LAKE primitive, specific to two relations $\mathcal{R}_a$ and $\mathcal{R}_b$, allows two users, Alice and Bob, owning a word $\mathrm{W}_a \in \mathrm{L}_{\mathcal{R}_a}(\mathsf{pub}, \mathsf{priv}_a)$ and $\mathrm{W}_b \in \mathrm{L}_{\mathcal{R}_b}(\mathsf{pub}, \mathsf{priv}_b)$ respectively, to agree on a session key under some specific conditions: they first both agree on the public parameter pub, Bob will think about priv'_a for his expected value of priv_a, Alice will do the same with priv'_b for priv_b; eventually, if $\mathsf{priv}'_a = \mathsf{priv}_a$ and $\mathsf{priv}'_b = \mathsf{priv}_b$, and if they both know words in the languages, then the key agreement will succeed. In case of failure, no information should leak about the reason of failure, except the inputs did not satisfy the relations $\mathcal{R}_a$ or $\mathcal{R}_b$, or the languages were not consistent.

We stress that each LAKE protocol is specific to a pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ describing the way Alice and Bob will authenticate to each other. This pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ specifies the sets $\mathcal{P}_a$, $\mathcal{P}_b$ and $\mathcal{S}_a$, $\mathcal{S}_b$ (to which the private parameters and the words should respectively belong). Therefore, the formats of priv_a, priv_b and W_a and W_b are known in advance, but not their values. When $\mathcal{R}_a$ and $\mathcal{R}_b$ are clearly defined from the context (e.g., PAKE), we omit them in the notations. For example, these relations can formalize:

- Password authentication: The language is defined by $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, \mathrm{W}) = 1 \Leftrightarrow \mathrm{W} = \mathsf{priv}$, and thus $\mathsf{pub} = \emptyset$. The classical setting of PAKE requires the players A and B to use the same password W, and thus we should have $\mathsf{priv}_a = \mathsf{priv}'_b = \mathsf{priv}_b = \mathsf{priv}'_a = \mathrm{W}_a = \mathrm{W}_b$;

- Signature authentication: $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, \mathrm{W}) = 1 \Leftrightarrow \mathsf{Verif}(\mathsf{pub}_1, \mathsf{pub}_2, \mathrm{W}) = 1$, where $\mathsf{pub} = (\mathsf{pub}_1 = \mathsf{vk}, \mathsf{pub}_2 = \mathrm{M})$ and $\mathsf{priv} = \emptyset$. The word W is a signature of M valid under vk, both specified in pub;

- Credential authentication: we can consider any mix for vk and M in pub or priv, and even in W, for which the relation $\mathcal{R}$ verifies the validity of the signature. When M and vk are in priv or W, we achieve *affiliation-hiding* property.

In the two last cases, the parameter pub can thus consist of a message on which the user is expected to know a signature valid under vk: either the user knows the signing key and can generate the signature on the fly to run the protocol, or the user has been given signatures on some messages (credentials). As a consequence, we just assume that, after having publicly agreed on a common pub, the two players have valid words in the appropriate languages. The way they have obtained these words does not matter.

Following our generic construction, private elements will be committed using encryption schemes, derived from Cramer-Shoup's scheme, and will thus have to be first encoded as n-tuples of elements in a group $\mathbb{G}$. In the case of PAKE, authentication will check that a player knows an appropriate password. The relation is a simple equality test, and accepts for one word only. A random commitment (and thus of a random group element) will succeed with negligible probability. For signature-based authentication, the verification key can be kept secret, but the signature should be unforgeable and thus a random word W should quite unlikely satisfy the relation. We will often make this assumption on useful relations $\mathcal{R}$: for any pub, $\{(\mathsf{priv}, \mathrm{W}) \in \mathcal{P} \times \mathcal{S}, \mathcal{R}(\mathsf{pub}, \mathsf{priv}, \mathrm{W}) = 1\}$ is sparse (negligible) in $\mathcal{P} \times \mathcal{S}$, and *a fortiori* in the set $\mathbb{G}^n$ in which elements are first embedded.

Ideal Functionality

We generalize the Password-Authenticated Key Exchange functionality $\mathcal{F}_{\text{PAKE}}$ (first provided in [CHK+05] and presented earlier) to more complex languages: The players agree on a common secret key if and only if they own words that lie in the languages the partners have in mind. More precisely, after an agreement on pub between P_i and P_j (modeled here by the use of the split functionality, see below), player P_i uses a word W_i belonging to $\mathrm{L}_i = \mathrm{L}_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and it expects its partner P_j to use a word W_j belonging to the language $\mathrm{L}'_j = \mathrm{L}_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}'_j)$, and vice-versa for P_j and P_i. We assume relations $\mathcal{R}_i$ and $\mathcal{R}_j$ to be specified by the kind of protocol we study (PAKE, Verifier-based PAKE, secret handshakes, ...) and so the languages are defined by the additional parameters pub, priv_i and priv_j only: They both agree on the public part pub, to be possibly parsed in a different way by each player for each language according to the relations. Note however that the respective languages do not need to be the same or to use similar relations: authentication means could be totally different for the 2 players. The key exchange should succeed if and only if the two following pairs of equations hold: ($\mathrm{L}'_i = \mathrm{L}_i$ and $\mathrm{W}_i \in \mathrm{L}_i$) and ($\mathrm{L}'_j = \mathrm{L}_j$ and $\mathrm{W}_j \in \mathrm{L}_j$).

In the initial $\mathcal{F}_{\text{PAKE}}$ functionality [CHK+05], the adversary was given access to a $\mathsf{TestPwd}$-query, which modeled the on-line dictionary attack. But it is known since [BCL+05] that it is equivalent to use the split functionality model [BCL+05], generate the $\mathsf{NewSession}$-queries corresponding to the corrupted players and tell the adversary (on behalf of the corrupted player) whether the protocol should succeed or not. Both methods enable the adversary to try a credential for a player (on-line dictionary attack). The second method (that we use here) implies allowing $\mathcal{S}$ to ask $\mathsf{NewSession}$-queries on behalf of the corrupted player, and letting it to be aware of the success or failure of the protocol in this case: the adversary learns this information only when it plays on behalf of a player (corruption or impersonation attempt). This is any way an information it would learn at the end of the protocol. We insist that third parties will not learn whether the protocol succeeded or not, as required for secret handshakes. To this aim, the NewKey-query informs in this case the adversary whether the credentials are consistent with the languages or not. In addition, the split functionality model guarantees from the beginning which player is honest and which one is controlled by the adversary. This finally allows us to get rid of the $\mathsf{TestPwd}$-query. The $\mathcal{F}_{\text{LAKE}}$ functionality is presented in Figure 5.4 and the corresponding split functionality $s\mathcal{F}_{\text{LAKE}}$ in Figure 5.5, where the languages are formally described and compared using the pub and priv parts.

The security goal is to show that the best attack for the adversary is a basic trial execution with a credential of its guess or choice: the proof will thus consist in emulating any real-life attack by either a trial execution by the adversary, playing as an honest player would do, but with a credential chosen by the adversary or obtained in any way; or a denial of service, where the adversary is clearly aware that its behavior will make the execution fail.

Fig. 5.4 – Ideal Functionality $\mathcal{F}_{\mathsf{lake}}$

The functionality $\mathcal{F}_{\text{LAKE}}$ is parametrized by a security parameter k and a public parameter pub for the languages. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathrm{P}_1,\ldots,\mathrm{P}_n$ via the following queries:

- New Session: Upon receiving a $\mathsf{NewSession}$-query $(\mathsf{sid}, \mathrm{P}_i, \mathrm{P}_j, \mathrm{W}_i, \mathrm{L}_i = \mathrm{L}(\mathsf{pub}, \mathsf{priv}_i), \mathrm{L}'_j = \mathrm{L}(\mathsf{pub}, \mathsf{priv}'_j))$ from P_i,
  - If this is the first $\mathsf{NewSession}$-query with identifier sid, record the tuple $(\mathrm{P}_i, \mathrm{P}_j, \mathrm{W}_i, \mathrm{L}_i, \mathrm{L}'_j, \mathsf{initiator})$. Send $(\mathsf{NewSession}; \mathsf{sid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pub}, \mathsf{initiator})$ to $\mathcal{S}$ and P_j.
  - If this is the second $\mathsf{NewSession}$-query with identifier sid, and if there exists a record $(\mathrm{P}_j, \mathrm{P}_i, \mathrm{W}_j, \mathrm{L}_j, \mathrm{L}'_i, \mathsf{initiator})$, then record the tuple $(\mathrm{P}_j, \mathrm{P}_i, \mathrm{W}_j, \mathrm{L}_j, \mathrm{L}'_i, \mathsf{initiator}, \mathrm{W}_i, \mathrm{L}_i, \mathrm{L}'_j, \mathsf{receiver})$ and send the answer $(\mathsf{NewSession}; \mathsf{sid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pub}, \mathsf{receiver})$ to $\mathcal{S}$ and P_j.
- Key Computation: Upon receiving a query $(\mathsf{NewKey} : \mathsf{sid})$ from $\mathcal{S}$, if there is a record of the form $(\mathrm{P}_i, \mathrm{P}_j, \mathrm{W}_i, \mathrm{L}_i, \mathrm{L}'_j, \mathsf{initiator}, \mathrm{W}_j, \mathrm{L}_j, \mathrm{L}'_i, \mathsf{receiver})$ and this is the first NewKey-query for session sid, then
  - If $(\mathrm{L}'_i = \mathrm{L}_i$ and $\mathrm{W}_i \in \mathrm{L}_i)$ and $(\mathrm{L}'_j = \mathrm{L}_j$ and $\mathrm{W}_j \in \mathrm{L}_j)$, then pick a random key sk of length k and store $(\mathsf{sid}, \mathsf{sk})$. If one player is corrupted, send $(\mathsf{sid}, \mathsf{success})$ to the adversary.
  - Else, store $(\mathsf{sid}, \perp)$, and send $(\mathsf{sid}, \mathsf{fail})$ to the adversary if one player is corrupted.
- Key Delivery: Upon receiving a query $(\mathsf{SendKey} : \mathsf{sid}, \mathrm{P}_i, \mathsf{sk})$ from $\mathcal{S}$, then
  - If there is a record of the form $(\mathsf{sid}, \mathsf{sk}')$, then, if both players are uncorrupted, output $(\mathsf{sid}, \mathsf{sk}')$ to P_i. Otherwise, output $(\mathsf{sid}, \mathsf{sk})$ to P_i.
  - If there is a record of the form $(\mathsf{sid}, \perp)$, then pick a random key sk' of length k and output $(\mathsf{sid}, \mathsf{sk}')$ to P_i.

Fig. 5.5 – Split Functionality $s\mathcal{F}_{\mathsf{lake}}$

Given the functionality $\mathcal{F}_{\text{LAKE}}$, the split functionality $s\mathcal{F}_{\text{LAKE}}$ proceeds as follows:

- Initialization:
  - Upon receiving $(\mathsf{Init}, \mathsf{sid}, \mathsf{pub}_i)$ from party P_i, send $(\mathsf{Init}, \mathsf{sid}, \mathrm{P}_i, \mathsf{pub}_i)$ to the adversary.
  - Upon receiving a message $(\mathsf{Init}, \mathsf{sid}, \mathrm{P}_i, \mathrm{H}, \mathsf{pub}, \mathsf{sid}_\mathrm{H})$ from $\mathcal{S}$, where $\mathrm{H} = \{\mathrm{P}_i, \mathrm{P}_j\}$ is a set of party identities, check that P_i has already sent $(\mathsf{Init}, \mathsf{sid}, \mathsf{pub}_i)$ and that for all recorded $(\mathrm{H}', \mathsf{pub}', \mathsf{sid}_{\mathrm{H}'})$, either $\mathrm{H} = \mathrm{H}'$, $\mathsf{pub} = \mathsf{pub}'$ and $\mathsf{sid}_\mathrm{H} = \mathsf{sid}_{\mathrm{H}'}$ or H and H' are disjoint and $\mathsf{sid}_\mathrm{H} \neq \mathsf{sid}_{\mathrm{H}'}$. If so, record the pair $(\mathrm{H}, \mathsf{pub}, \mathsf{sid}_\mathrm{H})$, send $(\mathsf{Init}, \mathsf{sid}, \mathsf{sid}_\mathrm{H}, \mathsf{pub})$ to P_i, and invoke a new functionality $(\mathcal{F}_{\text{LAKE}}, \mathsf{sid}_\mathrm{H}, \mathsf{pub})$ denoted as $\mathcal{F}_{\text{LAKE}}^{(\mathrm{H},\mathsf{pub})}$ and with set of honest parties H.
- Computation:
  - Upon receiving $(\mathsf{Input}, \mathsf{sid}, m)$ from party P_i, find the set H such that $\mathrm{P}_i \in \mathrm{H}$, the public value pub recorded, and forward m to $\mathcal{F}_{\text{LAKE}}^{(\mathrm{H},\mathsf{pub})}$.
  - Upon receiving $(\mathsf{Input}, \mathsf{sid}, \mathrm{P}_j, \mathrm{H}, m)$ from $\mathcal{S}$, such that $\mathrm{P}_j \notin \mathrm{H}$, forward m to $\mathcal{F}_{\text{LAKE}}^{(\mathrm{H},\mathsf{pub})}$ as if coming from P_j.
  - When $\mathcal{F}_{\text{LAKE}}^{(\mathrm{H},\mathsf{pub})}$ generates an output m for party $\mathrm{P}_i \in \mathrm{H}$, send m to P_i. If the output is for $\mathrm{P}_j \notin \mathrm{H}$ or for the adversary, send m to the adversary.

5.3.2 A Generic UC-Secure LAKE Construction

Intuition

Using smooth projective hash functions on commitments, one can generically define a LAKE protocol as done in [**ACP09**]. The basic idea is to make the player commit to their private information (for the expected languages and the owned words), and eventually the smooth projective hash functions will be used to make implicit validity checks of the global relation.

Details on commitments and associated smooth projective hash functions can be found in Sections 3 and 4 of [**BBC$^+$13a**], pages 99 and 101. The relations on the committed values will not be explicitly checked, since the values will never be revealed, but will be implicitly checked using SPHF. It is interesting to note that in both cases considered (one-part or two-part commitment), the projection key will only depend on the first part of the commitment.

As it is often the case in the UC setting, we need the initiator to use stronger primitives than the receiver. They both have to use non-malleable and extractable commitments, but the initiator will use a commitment that is additionally equivocable.

As already explained, SPHF will be used to implicitly check whether ($\mathrm{L}'_i = \mathrm{L}_i$ and $\mathrm{W}_i \in \mathrm{L}_i$) and ($\mathrm{L}'_j = \mathrm{L}_j$ and $\mathrm{W}_j \in \mathrm{L}_j$). But since in our instantiations private parameters priv and words W will have to be committed, the structure of these commitments will thus be publicly known in advance: commitments of $\mathcal{P}$-elements and $\mathcal{S}$-elements. Section 5.3.3 discusses on the languages captured by our definition, and illustrates with some AKE protocols. However, while these $\mathcal{P}$ and $\mathcal{S}$ sets are embedded in $\mathbb{G}^n$ from some n, it might be important to prove that the committed values are actually in $\mathcal{P}$ and $\mathcal{S}$ (e.g., one can have to prove it commits bits, whereas messages are first embedded as group elements in $\mathbb{G}$ of large order p). This will be an additional language-membership to prove on the commitments.

This leads to a very simple protocol described on Figure 5.6 (the notations for (double) linear Cramer-Shoup commitments LCS and DLCS can be found in Section 3 of the paper, page 99). Note that if a player wants to make external adversaries think it owns an appropriate word, as it is required for Secret Handshakes, he can still play, but will compute everything with dummy words, and will replace the ProjHash evaluation by a random value, which will lead to a random key at the end.

Security Analysis

Since we have to assume common pub, we make a first round (with flows in each direction) where the players send their contribution, to come up with pub. These flows will also be used to know if there is a player controlled by the adversary (as with the split functionality [BCL$^+$05]). In case the languages have empty pub, these additional flows are not required, since the split functionality can be applied on the committed values. The signing key for the receiver is not required anymore since there is one flow only from its side.

Theorem 3 ([**BBC$^+$13a**])**.** *Our LAKE scheme from Figure 5.6 realizes the $s\mathcal{F}_{\text{LAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$-hybrid model, in the presence of static adversaries, under the DLin assumption and the security of the One-Time Signature.*

Actually, from a closer look at the full proof (page 126), one can notice that $\mathsf{Com}_j = \mathcal{C}_j$ needs to be extractable, but IND-CPA security is enough, which leads to a shorter ciphertext (2 group elements less if one uses a Linear ciphertext instead of LCS). Similarly, one will not have to extract W_i from $\mathcal{C}_i$ when simulating sessions where P_i is corrupted. As a consequence, only the private parts of the languages have to be committed to in Com_i in the first and third rounds, whereas W_i can be encrypted independently with an IND-CPA encryption scheme in the third round only (5 group elements less in the first round, and 2 group elements less in the third round if one uses a Linear ciphertext instead of LCS).

Improvement

In [**BBC$^+$13b**], our new constructions of KV-SPHF enable several efficient instantiations of *one-round* LAKE protocols (page 147). The one-round PAKE scheme given in this paper is actually a particular case of a more general one-round LAKE scheme, for which we provide a BPR-like security model and a security proof (page 155).

Fig. 5.6 – LAKE from a Smooth Projective Hash Function on Commitments

Execution between P_i and P_j, with session identifier sid.

- Preliminary Round: each user generates a pair of signing/verification keys $(\mathsf{SK}, \mathsf{VK})$ and sends VK together with its contribution to the public part of the language.

We denote by ℓ_i the label $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pub}, \mathsf{VK}_i, \mathsf{VK}_j)$ and by ℓ_j the label $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, \mathsf{pub}, \mathsf{VK}_j, \mathsf{VK}_i)$, where pub is the combination of the contributions of the two players. The initiator now uses a word W_i in the language $\mathrm{L}(\mathsf{pub}, \mathsf{priv}_i)$, and the receiver uses a word W_j in the language $\mathrm{L}(\mathsf{pub}, \mathsf{priv}_j)$, possibly re-randomized from their long-term secrets[*]. We assume commitments and associated smooth projective hash functions exist for these languages.

- First Round: user P_i (with random tape ω_i) generates a multi-$\mathsf{DLCSCom}'$ commitment on $(\mathsf{priv}_i, \mathsf{priv}'_j, \mathrm{W}_i)$ in $(\mathcal{C}_i, \mathcal{C}'_i)$, where W_i has been randomized in the language, under the label ℓ_i. It also computes a Pedersen commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with random exponent t). It then sends $(\mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
- Second Round: user P_j (with random tape ω_j) computes a multi-LCS commitment on $(\mathsf{priv}_j, \mathsf{priv}'_i, \mathrm{W}_j)$ in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\vec{r}$, where W_j has been randomized in the language, under the label ℓ_j. It then generates a challenge $\vec{\varepsilon}$ on $\mathcal{C}_i$ and hashing/projection keys[†] hk_i and hp_i associated to $\mathcal{C}_i$ (which will be associated to the future Com_i). It finally signs all the flows using SK_j in σ_j, and sends $(\mathcal{C}_j, \vec{\varepsilon}, \mathsf{hp}_i, \sigma_j)$ to P_i;
- Third Round: user P_i first checks the signature σ_j, computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\vec{\varepsilon}}_i$ and witness $\mathbf{z}$ (from $\vec{\varepsilon}$ and ω_i), it generates hashing/projection keys hk_j and hp_j associated to Com_j. It finally signs all the flows using SK_i in σ_i, and sends $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to P_j;
- Hashing: P_j first checks the signature σ_i and the correct opening of $\mathcal{C}''_i$ into $\mathcal{C}'_i$, it computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\vec{\varepsilon}}_i$. P_i computes K_i and P_j computes K_j as follows:

$$\begin{aligned}
\mathrm{K}_i &= \mathsf{Hash}(\mathsf{hk}_j, \{(\mathsf{priv}'_j, \mathsf{priv}_i)\} \times \mathrm{L}(\mathsf{pub}, \mathsf{priv}'_j), \ell_j, \mathsf{Com}_j) \\
&\quad \times \mathsf{ProjHash}(\mathsf{hp}_i, \{(\mathsf{priv}_i, \mathsf{priv}'_j)\} \times \mathrm{L}(\mathsf{pub}, \mathsf{priv}_i), \ell_i, \mathsf{Com}_i; \mathbf{z}) \\
\mathrm{K}_j &= \mathsf{ProjHash}(\mathsf{hp}_j, \{(\mathsf{priv}_j, \mathsf{priv}'_i)\} \times \mathrm{L}(\mathsf{pub}, \mathsf{priv}_j), \ell_j, \mathsf{Com}_j; \vec{r}) \\
&\quad \times \mathsf{Hash}(\mathsf{hk}_i, \{(\mathsf{priv}'_i, \mathsf{priv}_j)\} \times \mathrm{L}(\mathsf{pub}, \mathsf{priv}'_i), \ell_i, \mathsf{Com}_i)
\end{aligned}$$

[*]As explained in Section 5.3.1, recall that the languages considered depend on two possibly different relations, namely $\mathrm{L}_i = \mathrm{L}_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and $\mathrm{L}_j = \mathrm{L}_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}_j)$, but we omit them for the sake of clarity. We assume they are both self-randomizable.

[†]Recall that the SPHF is constructed in such a way that this projection key does not depend on $\mathcal{C}'_i$ and is indeed associated to the future whole Com_i.

5.3.3 Concrete Instantiations and Comparisons

Our generic protocol of LAKE enables us to give in [**BBC⁺13a**] some concrete instantiations of several AKE protocols, such as PAKE, verifier-based PAKE, CAKE and secret handshakes. We here give the main ideas for the two first and refer the reader to the article for the two others (page 108).

Password-Authenticated Key Exchange

Using our generic construction, we can easily obtain a PAKE protocol, as described on Figure 5.7, where we optimize from the generic construction, since $\mathsf{pub} = \emptyset$, removing the agreement on pub, but still keeping the one-time signature keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ to avoid man-in-the-middle attacks since it has another later flow: P_i uses a password W_i and expects P_j to own the same word, and thus in the language $\mathrm{L}'_j = \mathrm{L}_i = \{\mathrm{W}_i\}$; P_j uses a password W_j and expects P_i to own the same word, and thus in the language $\mathrm{L}'_i = \mathrm{L}_j = \{\mathrm{W}_j\}$; The relation is the equality test between priv_i and priv_j, which both have no restriction in $\mathbb{G}$ (hence $\mathcal{P} = \mathbb{G}$). As the word W_i, the language private parameters priv_i of a user and priv'_j of the expected language for the other user are the same, each user can commit in the protocol to only one value: its password.

Fig. 5.7 – Password-based Authenticated Key Exchange

P_i uses a password W_i and P_j uses a password W_j. We denote $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$.

- First Round: P_i (with random tape ω_i) first generates a pair of signing/verification keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ and a $\mathsf{DLCSCom}'$ commitment on W_i in $(\mathcal{C}_i, \mathcal{C}'_i)$, under $\ell_i = (\ell, \mathsf{VK}_i)$. It also computes a Pedersen commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with random exponent t). It then sends $(\mathsf{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
- Second Round: P_j (with random tape ω_j) computes a LCSCom commitment on W_j in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\vec{r}$, under the label ℓ. It then generates a challenge ε on $\mathcal{C}_i$ and hashing/projection keys hk_i and the corresponding hp_i for the equality test on Com_i ("Com_i is a valid commitment of W_j", this only requires the value ξ_i computable thanks to $\mathcal{C}_i$). It then sends $(\mathcal{C}_j, \varepsilon, \mathsf{hp}_i)$ to P_i;
- Third Round: user P_i can compute $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$ and witness $\mathbf{z}$ (from ε and ω_i), it generates hashing/projection keys hk_j and hp_j for the equality test on Com_j. It finally signs all the flows using SK_i in σ_i and sends $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to P_j;
- Hashing: P_j first checks the signature and the validity of the Pedersen commitment (thanks to t), it computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$. P_i computes K_i and P_j computes K_j as follows:

$$\mathrm{K}_i = \mathsf{Hash}(\mathsf{hk}_j, \mathrm{L}'_j, \ell, \mathsf{Com}_j) \cdot \mathsf{ProjHash}(\mathsf{hp}_i, \mathrm{L}_i, \ell_i, \mathsf{Com}_i; \mathbf{z})$$
$$\mathrm{K}_j = \mathsf{ProjHash}(\mathsf{hp}_j, \mathrm{L}_j, \ell, \mathsf{Com}_j; \vec{r}) \cdot \mathsf{Hash}(\mathsf{hk}_i, \mathrm{L}'_i, \ell_i, \mathsf{Com}_i)$$

Verifier-based PAKE

The above scheme can be modified into an efficient PAKE protocol that is additionally secure against *server compromise*: the so-called verifier-based PAKE, where the client owns a password pw, while the server knows a verifier only, such as g^{pw}, so that in case of break-in to the server, the adversary will not immediately get all the passwords.

To this aim, as usually done, one first does a PAKE with g^{pw} as common password, then asks the client to additionally prove it can compute the Diffie-Hellman value h^{pw} for a basis h chosen by the server. Ideally, we could implement this trick, where the client P_j just considers the equality test between the g^{pw} and the value committed by the server for the language $\mathrm{L}'_i = \mathrm{L}_j$, while the server P_i considers the equality test with $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, where h is sent as its contribution to the public part of the language by the server $\mathrm{L}_i = \mathrm{L}'_j$. Since the server chooses h itself, it chooses it as $h = g^\alpha$, for an ephemeral random α, and can thus compute $h^{\mathsf{pw}} = (g^{\mathsf{pw}})^\alpha$. On its side, the client can compute this value since it knows pw. The client could thus commit to $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, in order to prove its knowledge of pw, whereas the server could just commit to g^{pw}. Unfortunately, from the extractability of the server commitment, one would just get g^{pw}, which is not enough to simulate the client.

To make it in a provable way, the server chooses an ephemeral h as above, and they both run the previous PAKE protocol with $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$ as common password, and mutually checked: h is seen as the pub part, hence the preliminary flows are required.

5.4 Extension to Distributed PAKE

5.4.1 Definition and Security Model

Introduction

Incidents of sensitive customer information "hacking" (including leaking of passwords) in e-commerce systems are frequently revealed in the newspaper. In addition to major reputational damage, a company with a significant data breach may be sued by its clients for the breach and may be suspended or disqualified from future public sector or government work.

To alleviate the threat that stored passwords are revealed immediately in case of a server compromise, many servers adopt the approach for storing passwords in a hashed form with a random salt. When the database of hashed password is compromised, the offline dictionary attack requires a more important computational effort but remains usually possible. The notion of *Verifier-based PAKE*, where the client owns a password pw and the server knows a one-way transformation v of the password only, was proposed by Bellovin and Merritt [BM92]. The two players eventually agree on a common high entropy secret if and only if pw and v match together. It prevents massive password recovering in case of server corruption and it forces the attacker who breaks into the server and is willing to recover passwords to perform an additional costly offline dictionary attack.

We consider in **[BCV16]** an alternative approach inspired by the multi-party computation paradigm (and first suggested by Ford and Kaliski [FK00]). The password database on the server side is somehow shared among two servers (or more, but we focus here on two for sake of simplicity), and authentication requires a distributed computation involving the client – who still does not need an additional cryptographic device capable of storing high-entropy secret keys – and the two servers who will use some additional shared secret information. The interaction is performed using a *gateway* that does not know any secret information and ends up in the gateway and the client sharing a common key. The lifetime of the protocol is divided into distinct periods (for simplicity, one may think of these time periods as being of equal length; e.g. one day) and at the beginning of each period, the two servers interact and update their sharing of the password database. Similarly to proactive schemes in multi-party computation, we allow the adversary multiple corruptions of each server, limiting only the corruptions to one server for each period. The user does not need to update his password nor to perform any kind of computations and its interaction with the two servers (performed using the gateway) remains the same for the lifetime of the protocol. In this scenario, even if a server compromise is doable, the secret exposure is not valuable to the adversary since it reveals only a share of the password database and does not permit to run an offline dictionary attack.

Ford and Kaliski [FK00] were the first to propose to distribute the capability to test passwords over multiple servers. Building on this approach, several such protocols were subsequently proposed in various settings (for instance, [Jab01, MSJ02, BJKS03, DG06, SK05, KMTG12, ACFP05, KM14]) and it is worth noting that the protocol from [BJKS03] is commercially available as EMC's *RSA Distributed Credential Protection*. Recently, Camenisch, Enderlein and Neven [CEN15] revisited this approach and proposed a scheme in the universal composability framework [Can01] (which has obvious advantages for password-based protocols since users often use related passwords for many providers). Camenisch *et al.* gave interesting details about the steps that need to be taken when a compromise actually occurs. Unfortunately, due to the inherent difficulties of construction of the simulator in the universal composability framework, their scheme is inefficient since users and servers have to perform a few hundred exponentiations each.

Our Constructions

We present in our paper practical realizations based on classical cryptographic assumptions in the standard security model. In order to achieve practical constructions in the standard security model, we consider the variant of the BPR model[2] in the distributed setting proposed by Katz, MacKenzie, Taban and Gligor in [KMTG12]. In this security model, we assume that the communication between the client and the authentication servers is carried on a basically insecure network. Messages can be tapped and modified by an adversary and the communication between the clients and the servers is asynchronous. The adversary should not be able to brute force guess a password without further interactions with the client for each guess even if he corrupts and impersonates a server in an active way.

Our first construction uses a similar approach to the schemes from [Jab01, MSJ02, BJKS03, DG06, SK05, KMTG12, ACFP05, KM14]: The user generates information theoretic shares of his password and sends them to the servers. In the authentication phase, the parties run a dedicated protocol to verify that the provided password equals the priorly shared one. Our solution then consists in some sort of three-party PAKE, in which (1) the user implicitly checks (using a smooth projective hash function) that its password is indeed the sum of the shares owned by the two servers, and (2) each server implicitly checks that its share is the difference of the password owned by the user and the share owned by the other server. Contrary to the popular approach initiated in [KOY01, GL03] for PAKE, we cannot use two smooth projective hash functions (one for the client and one for the server) so we propose a technique in order to combine in a secure way six smooth projective hash functions. This new method (which may be

[2]Our schemes can be adapted to achieve security in universal composability framework using techniques similar to those used in [CEN15]. The resulting schemes are slightly more efficient but are unfortunately still not practical.

of independent interest) allows us to prove the security of this construction under classical cryptographic assumptions (namely the DDH assumption) in the standard security model from [KMTG12] (without any idealized assumptions).

The main weakness of this first solution is that at each time period, the servers have to refresh the information-theoretic sharing of the password of all users. This can be handled easily using well-known techniques from proactive multi-party computation but if the number of users is large, this can be really time-consuming (in particular if the time period is very short).

Our second construction is built on the ideas from the first one but passwords are now encrypted using a public-key encryption scheme where the corresponding secret key is shared among the servers. At the beginning of each time period, the servers only need to refresh the sharing of this secret key but the password database is not modified (and can actually be public). Password verification and the authenticated key exchange is then carried out without ever decrypting the database. A secure protocol is run to verify that the password sent by the user matches the encrypted password. It is similar to the protocol we design for the first construction except that the user encrypts its password and the parties implicitly check (using in this case five smooth projective hash functions) that the message encrypted in this ciphertext is the same as the message encrypted in the database (using the secret key shared upon the servers). Both constructions consist in only two flows (one from the client and one from the servers) and a (private) flow from the servers to the gateway.

Security Model

In a distributed PAKE system, we consider as usual a client (owning a password) willing to interact with a gateway, such as a website. The difference compared to a non-distributed system is that the gateway itself interacts with two servers, and none of the three owns enough information to be able to recover the passwords of the clients on its own[3]. Such a scheme is correct if the interaction between a client with a correct password and the gateway succeeds. An honest execution of a distributed PAKE protocol should result in the client holding a session key K_U and the gateway holding a session key $K_G = K_U$.

We propose here two settings that describe well this situation. In a first setting, we consider that the passwords of the clients are shared information-theoretically between the servers, such as $\pi = \pi_1 + \pi_2$ (if the password π belongs to an appropriate group) or with the help of any secret sharing protocol. At the beginning of each time period, the shares are updated, in a probabilistic way, using a public function Refresh, depending on the sharing protocol used.

In a second setting, we consider that the gateway owns a database of encrypted passwords (which can be considered public), and the servers each own a share of the corresponding private keys (obtained by a secret sharing protocol). At the beginning of each time period, the shares are updated, in a probabilistic way, using a public function Refresh, depending on the sharing protocol used.

Since the security of our schemes is not analyzed in the universal composability framework (contrary to the recent paper [CEN15]), the Refresh procedure can be handled easily using classical techniques from computational proactive secret sharing (see [OY91, HJKY95] for instance).

We consider the classical model [BPR00] for authenticated key-exchange, adapted to the two-server setting by [ACFP05, KMTG12]. In the latter model, the authors assume that every client in the system shares its password with exactly two servers. We loosen this requirement here, depending on the setting considered, as described above. We refer the interested reader to these articles for the details and we give the high-level ideas in [**BCV16**].

5.4.2 Our Simple Protocol

Description of the Setting

In this first setting, we consider a client U owning a password π and willing to interact with a gateway G. The gateway interacts with two servers S_1 (owning π_1) and S_2 (owning π_2), such that $\pi = \pi_1 + \pi_2$. It should be noted that only the client's password is assumed to be small and human-memorable. The two "passwords" owned by the servers can be arbitrarily big. The aim of the protocol is to establish a shared session key between the client and the gateway.

A simple solution to this problem consists in considering some sort of three-party PAKE, in which the client implicitly checks (using an SPHF) whether its password is the sum of the shares owned by the two servers, and the servers implicitly check (also using an SPHF) whether their share is the difference of the

[3]Note that the gateway can be merged with one server.

password owned by the client and the share owned by the other server. For sake of simplicity, we denote the client U as S_0 and its password π as π_0.

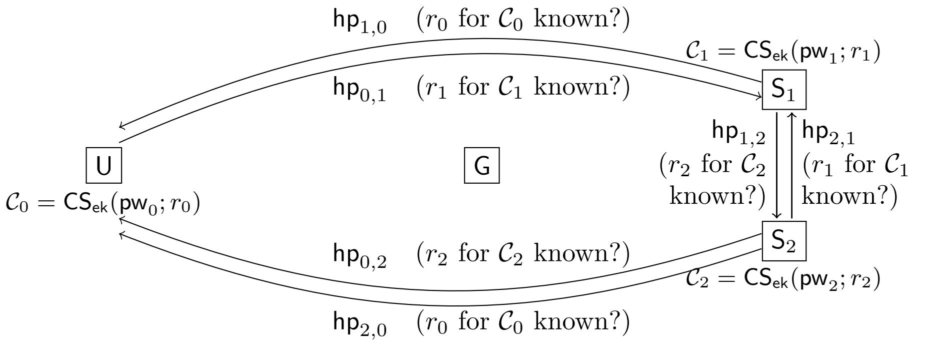
Main Idea of the Construction

In our setting, we denote by $\mathsf{pw}_b = g^{\pi_b}$. The main idea of the protocol is depicted on Figure 5.8. For sake of readability, the participants which have a real role in the computations are directly linked by arrows in the picture, but one should keep in mind that all the participants (U, S_1 and S_2) only communicate with G, which then broadcasts all the messages.

In a classical SPHF-based two-party key-exchange between U and G, the client and the gateway would compute a Cramer-Shoup encryption of their password: $\mathcal{C}_0 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_0; r_0)$ and $\mathcal{C}_{\mathsf{G}} = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_{\mathsf{G}}; r_{\mathsf{G}})$. The gateway would then send a projection key $\mathsf{hp}_{\mathsf{G},0}$ in order to implicitly check via an SPHF whether $\mathcal{C}_0$ is a valid Cramer-Shoup encryption of pw_{G}, and the client would send a projection key $\mathsf{hp}_{0,\mathsf{G}}$ in order to implicitly check via an SPHF whether $\mathcal{C}_{\mathsf{G}}$ is a valid Cramer-Shoup encryption of pw_0.

Here, since S_0 owns $\mathsf{pw}_0 = \mathsf{pw}_1 \cdot \mathsf{pw}_2$, so that the players do not share the same password, we consider an SPHF between each pair of players $(\mathcal{S}_i, \mathcal{S}_j)$, in which player $\mathcal{S}_i$ computes the ciphertext $\mathcal{C}_i = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_i; r_i)$, the keys $\mathsf{hk}_{i,j}$ and $\mathsf{hp}_{i,j}$ and sends $(\mathcal{C}_i, \mathsf{hp}_{i,j})$ to $\mathcal{S}_j$. It also computes the hash value $\mathrm{H}_{i,j} = \mathsf{Hash}(\mathsf{hk}_{i,j}, (\mathsf{ek}, \mathsf{pw}_i), \mathcal{C}_j)$ and the projected hash value $\mathrm{H}'_{j,i} = \mathsf{ProjHash}(\mathsf{hp}_{j,i}, (\mathsf{ek}, \mathrm{M}_i), \mathcal{C}_i, r_i)$. Formally, for each pair of users $(\mathcal{S}_i, \mathcal{S}_j)$, the language checked on $\mathcal{S}_j$ by $\mathcal{S}_i$ is defined as follows: $\mathcal{C}_j \in \mathcal{L}_{i,j} = \{\mathcal{C} = (u_1, u_2, e, v) \in \mathbb{G}^4 \mid \exists r \in \mathbb{Z}_p \text{ such that } \mathcal{C} = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_j; r)\}$ but it cannot be checked directly by a unique SPHF since the passwords are different (and thus $\mathcal{S}_i$ does not know pw_j). Rather, we combine in the protocol the six SPHF described to globally ensure the correctness (each one remaining smooth and pseudo-random), as described in the next part. The correctness of the SPHF for the pair $(\mathcal{S}_i, \mathcal{S}_j)$ implies that if everything was computed honestly, then one gets the equalities $\mathrm{H}_{i,j}(\mathsf{pw}_i/\mathsf{pw}_j)^{\lambda_i} = \mathrm{H}'_{i,j}$ and $\mathrm{H}_{j,i}(\mathsf{pw}_j/\mathsf{pw}_i)^{\lambda_j} = \mathrm{H}'_{j,i}$.

Fig. 5.8 – Main idea of the construction (simple protocol)


$\mathsf{hp}_{1,0}$ (r_0 for $\mathcal{C}_0$ known?)
$\mathcal{C}_1 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_1; r_1)$
$\mathsf{hp}_{0,1}$ (r_1 for $\mathcal{C}_1$ known?)
S_1
$\mathsf{hp}_{1,2}$ (r_2 for $\mathcal{C}_2$ known?)
$\mathsf{hp}_{2,1}$ (r_1 for $\mathcal{C}_1$ known?)
U
G
$\mathcal{C}_0 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_0; r_0)$
S_2
$\mathsf{hp}_{0,2}$ (r_2 for $\mathcal{C}_2$ known?)
$\mathcal{C}_2 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_2; r_2)$
$\mathsf{hp}_{2,0}$ (r_0 for $\mathcal{C}_0$ known?)

5.4.3 Our Efficient Protocol

Description of the Setting

In this second setting, we consider again a client U owning a password π and willing to interact with a gateway G. The gateway owns a public database of encrypted passwords, and it interacts with two servers S_1 and S_2, each owning a share of the secret key of the encryption scheme. The aim of the protocol is to establish a shared session key between the client and the gateway.

The idea is similar to the protocol described in the former section, except that only the client needs to compute a ciphertext, the other ciphertext being publicly available from the database. The participants implicitly check (using several SPHF) that the message encrypted in the ciphertext of the client is the same as the message encrypted in the database (using the secret key shared upon the servers).

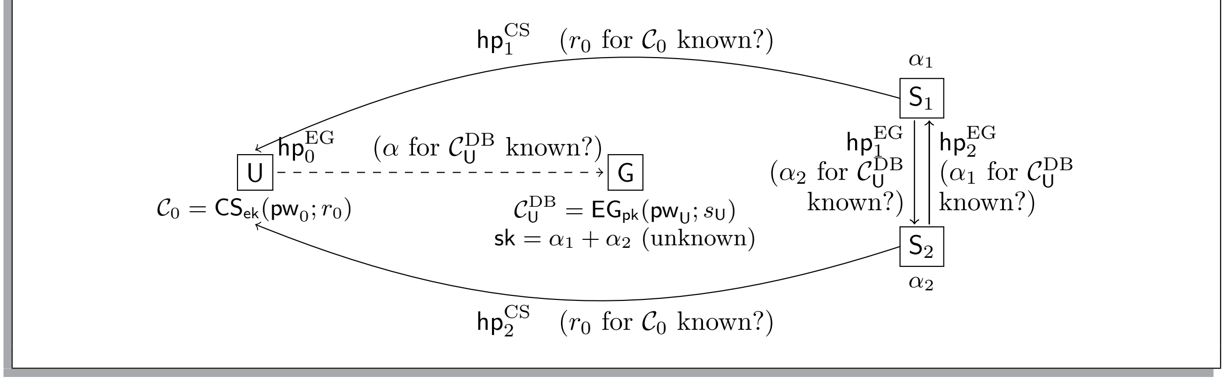
Main Idea of the Construction

Again, we denote the client U as S_0 and its password π as π_0. In our setting, we denote by $\mathsf{pw}_k = g^{\pi_k}$ for all k. The database contains El Gamal encryptions of each ciphertext $\mathsf{pw}_{\mathsf{U}_i}$, under randomness s_{U_i}:

$\mathcal{C}_{\mathsf{U}_i}^{\mathrm{DB}} = \mathsf{EG}_{\mathsf{pk}}(\mathsf{pw}_{\mathsf{U}_i}; s_{\mathsf{U}_i}) = (h^{s_{\mathsf{U}_i}}\mathsf{pw}_{\mathsf{U}_i}, g^{s_{\mathsf{U}_i}})$, so that here, $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}} = \mathsf{EG}_{\mathsf{pk}}(\mathsf{pw}_{\mathsf{U}}; s_{\mathsf{U}}) = (h^{s_{\mathsf{U}}}\mathsf{pw}_{\mathsf{U}}, g^{s_{\mathsf{U}}})$. The client computes a Cramer-Shoup encryption of its password: $\mathcal{C}_0 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_0; r_0) = (u_1, u_2, e, v)$ with $v = cd^{\xi}$. The execution of the protocol should succeed if these encryptions are correct and $\mathsf{pw}_0 = \mathsf{pw}_{\mathsf{U}}$. Recall that the server $\mathcal{S}_i$ knows α_i such that $\alpha = \alpha_1 + \alpha_2$ is the decryption key of the El Gamal encryption.

The main idea is depicted on Figure 5.9. For sake of readability, the participants which have a real role in the computations are directly linked by arrows in the picture, but one should keep in mind that all the participants (U, S_1 and S_2) only communicate with G, which then broadcasts all the messages.

Fig. 5.9 – Main idea of the construction (efficient protocol)


$\mathsf{hp}_1^{\mathrm{CS}}$ (r_0 for $\mathcal{C}_0$ known?)
α_1
S_1
$\mathsf{hp}_0^{\mathrm{EG}}$ (α for $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ known?)
U
G
$\mathsf{hp}_1^{\mathrm{EG}}$ (α_2 for $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ known?)
$\mathsf{hp}_2^{\mathrm{EG}}$ (α_1 for $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ known?)
$\mathcal{C}_0 = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_0; r_0)$
$\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}} = \mathsf{EG}_{\mathsf{pk}}(\mathsf{pw}_{\mathsf{U}}; s_{\mathsf{U}})$
$\mathsf{sk} = \alpha_1 + \alpha_2$ (unknown)
S_2
α_2
$\mathsf{hp}_2^{\mathrm{CS}}$ (r_0 for $\mathcal{C}_0$ known?)

In a classical SPHF-based two-party key-exchange between the user U and the gateway G, the gateway would check whether $\mathcal{C}_0$ is a valid Cramer-Shoup encryption of pw_{U}. Since here the password pw_{U} is unknown to the servers S_1 and S_2, this is done in our setting by two SPHF, using $\mathsf{hp}_1^{\mathrm{CS}}$ (sent by S_1) and $\mathsf{hp}_2^{\mathrm{CS}}$ (sent by S_2), where the servers use the first term of the public encryption $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ ($h^{s_{\mathsf{U}}}\mathsf{pw}_{\mathsf{U}}$) in order to cancel the unknown pw_{U}.

In a classical SPHF-based two-party key-exchange between U and G, the client would also check whether $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ is a valid El Gamal encryption of its password pw_0, i.e. whether the gateway knows a witness for its ciphertext $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ (s_{U} in the usual constructions, α here). Since α is unknown to the gateway, this is done in our setting by the combination of three SPHF, using $\mathsf{hp}_0^{\mathrm{EG}}$ (sent by the client), $\mathsf{hp}_1^{\mathrm{EG}}$ (sent by S_1) and $\mathsf{hp}_2^{\mathrm{EG}}$ (sent by S_2). These three SPHF allow the client and the servers to implicitly check that the servers know α_1 and α_2 such that $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ can be decrypted (using the decryption key $\alpha = \alpha_1 + \alpha_2$) to the same password pw_0 than the one encrypted in $\mathcal{C}_0$ sent by the client. Formally, the languages checked are as follows:

- by the client:
  $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}} \in \mathcal{L}_0 = \{\mathcal{C} = (e, u) \in \mathbb{G}^2 \mid \exists \alpha \in \mathbb{Z}_p \text{ such that } h = g^{\alpha} \text{ and } e/u^{\alpha} = \mathsf{pw}_0\}$
- by server $\mathcal{S}_i$ (with respect to the client $\mathcal{S}_0$ and server $\mathcal{S}_j$):
  $\mathcal{C}_0 \in \mathcal{L}_{i,0} = \{\mathcal{C} = (u_1, u_2, e, v) \in \mathbb{G}^4 \mid \exists r \in \mathbb{Z}_p \text{ such that } \mathcal{C} = \mathsf{CS}_{\mathsf{ek}}(\mathsf{pw}_{\mathsf{U}}; r)$ and $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}} \in \mathcal{L}_{i,j} = \{\mathcal{C} = (e, u) \in \mathbb{G}^2 \mid \exists \alpha_j \in \mathbb{Z}_p \text{ such that } h = g^{\alpha_i + \alpha_j}\}$ and $e/u^{\alpha_i+\alpha_j} = \mathsf{pw}_{\mathsf{U}}\}$

but they cannot be checked directly by a unique SPHF since the value pw_{U} appearing in the languages is unknown to the verifier $\mathcal{S}_i$. Rather, the server $\mathcal{S}_i$ will use the first term of the public encryption $\mathcal{C}_{\mathsf{U}}^{\mathrm{DB}}$ ($h^{s_{\mathsf{U}}}\mathsf{pw}_{\mathsf{U}}$) in order to cancel this unknown pw_{U}. To achieve this goal, we combine the five SPHF described to globally ensure the correctness (each one remaining smooth and pseudo-randomness), as described in [**BCV16**].

Chapter 6

Oblivious Transfer

6.1 Definition and Security Model

In an oblivious transfer scheme, we consider the interaction between a server, possessing a database called DB containing $k = 2^d$ lines, and a user, willing to request the line s of the database in an oblivious way. Informally, this implies that the user will gain no information about the other lines of the database, and also that the server will obtain no information about the specific line the user wants to obtain.

The ideal functionality of an Oblivious Transfer (OT) protocol was given in [Can01, CKWZ13], [**ABB$^+$13**]. We recall it in Figure 6.1. We also give a version of this functionality [**BCG16**] in simple UC [CCL15] in Figure 6.2. The main difference is that there is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party).

The party P_i is the sender $\mathcal{S}$, while the party P_j is the receiver $\mathcal{R}$. The former is provided with a database consisting of a set of k lines $(m_1, \dots, m_k)$, while the latter is querying a particular line m_s (with $s \in \{1, \dots, k\}$). Since there is no communication between them (the functionality deals with everything), it automatically ensures the oblivious property on both sides (the sender does not learn which line was queried, while the receiver does not learn any line other than L_s).

Fig. 6.1 – Ideal Functionality for 1-out-of-k Oblivious Transfer $\mathcal{F}_{(1,k)\text{-OT}}$

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathrm{P}_1,\dots,\mathrm{P}_n$ via the following queries:

- **Upon receiving an input $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \dots, m_k))$ from party P_i**, with $(m_1, \dots, m_k) \in (\{0,1\}^{\mathfrak{K}})^k$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \dots, m_k))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.

- **Upon receiving an input $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, s)$ from party P_j,** with $s \in \{1, \dots, k\}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, s)$, and reveal $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to the adversary $\mathcal{S}$. Ignore further $\mathsf{Receive}$-message with the same ssid from P_j.

- **Upon receiving a message $(\mathsf{Sent}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \dots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, s)$ is not recorded; otherwise send $(\mathsf{Sent}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to P_i and ignore further Sent-message with the same ssid from the adversary.

- **Upon receiving a message $(\mathsf{Received}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \dots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, s)$ is not recorded; otherwise send $(\mathsf{Received}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, m_s)$ to P_j and ignore further $\mathsf{Received}$-message with the same ssid from the adversary.

Fig. 6.2 – Ideal Functionality for 1-out-of-k Oblivious Transfer $\mathcal{F}_{(1,k)\text{-OT}}$ (simple UC)

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties P_1,...,P_k via the following queries:

- **Upon receiving an input $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \ldots, m_k))$ from party P_i**, with $(m_1, \ldots, m_k) \in (\{0,1\}^{\mathfrak{K}})^k$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \ldots, m_k))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.

- **Upon receiving an input $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, s)$ from party P_j**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1, \ldots, m_k))$ is not recorded. Otherwise, reveal $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to the adversary $\mathcal{S}$ and send $(\mathsf{Received}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, m_s)$ to P_j and ignore further Receive-message with the same ssid from P_j.

6.2 Constructions

6.2.1 Three-round adaptively secure 1-out-of-k OT

In [**ABB+13**], we provide a generic construction of a three-round UC-secure 1-out-of-k OT (see page 185) from any SPHF-friendly commitment (see definition in Section 4.3.3). The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS. Besides decreasing the total number of rounds with respect to existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solution so far [CKWZ13]. Moreover, our construction is more general and provides a solution for 1-out-of-k OT schemes while the solution in [CKWZ13] only works for $k = 2$.

Generic Construction from SPHF-Friendly Commitments

In Figure 6.3, we describe a 3-round OT that is UC-secure against adaptive adversaries, and a 2-round variant which is UC-secure against static adversaries. They can be built from any SPHF-friendly commitment scheme, where $\mathfrak{L}_t$ is the language of the commitments that open to t under the associated label ℓ, and from any IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with plaintext size at least $\mathfrak{K}$, and from any Pseudo-Random Generator (PRG) F with input size equal to plaintext size, and output size equal to the size of the messages in the database. Notice the adaptive version can be seen as a variant of the static version where the last flow is sent over a somewhat secure channel, as in [CKWZ13]; and the preflow and pk and c are used to create this somewhat secure channel.

Theorem 4 ([**ABB+13**]). *The two Oblivous Transfer schemes described in Figure 6.3 are UC-secure in the presence of adaptive adversaries and static adversaries respectively, assuming reliable erasures and authenticated channels, as soon as the commitment scheme is SPHF-friendly.*

Concrete Instantiation and Comparison

Using our commitment $\mathcal{E}^2\mathcal{C}$ along with its SPHF, one gets the protocol described in [**ABB+13**] (page 205), where the number of bits of the commited value is $d = \lceil \log k \rceil$. The comparison of the communication costs with other schemes (in particular [CKWZ13]) can be found in Figure 6.10 page 74.

6.2.2 Generic Construction

In [**BC15**], our generic construction of SPHF-friendly commitment scheme (see Section 4.3.3) allows us to provide a generic way to obtain a UC-secure OT scheme from the same building blocks (CH and CCA encryption) and three concrete instantiations from DDH, LWE and DCR. This construction is generic and does not specifically induce pairings (as in [**ABB+13**]). Furthermore, our 3 instantiations come straightforward from our generic framework (and [**ABB+13**] can be derived from it). The complexity comparisons can be found in Figure 6.10 page 74.

Fig. 6.3 – UC-Secure 1-out-of-k OT from an SPHF-Friendly Commitment

CRS generation:
$\rho \stackrel{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param} \stackrel{\$}{\leftarrow} \mathsf{Setup}(1^{\mathfrak{K}})$.

Pre-flow (for adaptive security only)**:**
1. P_i generates a key pair $(\mathsf{pk}, \mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{KeyGen}(\mathsf{param})$ for $\mathcal{E}$
2. P_i stores sk, completely erase random coins used by KeyGen, and sends pk to P_i

Index query on s:
1. P_j chooses a random value S, computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{S})$ and encrypts S under pk: $c \stackrel{\$}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}, \mathrm{S})$ (for adaptive security only; for static security: $c = \perp, \mathrm{R} = 0$)
2. P_j computes $(\mathrm{C}, \delta) \stackrel{\$}{\leftarrow} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$
3. P_j stores δ, completely erases S and random coins used by Com and $\mathsf{Encrypt}$, and sends C and c to P_i

Database input $(m_1, \ldots, m_k)$:
1. P_i decrypts $\mathrm{S} \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$ and gets $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{S})$ (for static security: $\mathrm{R} = 0$)
2. P_i computes $\mathsf{hk}_t \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathfrak{L}_t)$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, \mathfrak{L}_t, (\ell, \mathrm{C}))$,
$\mathrm{K}_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, \mathfrak{L}_t, (\ell, \mathrm{C}))$, and $\mathrm{M}_t \leftarrow \mathrm{R} \oplus \mathrm{K}_t \oplus m_t$, for $t = 1, \ldots, k$
3. P_i erases everything except $(\mathsf{hp}_t, \mathrm{M}_t)_{t=1,\ldots,k}$ and sends $(\mathsf{hp}_t, \mathrm{M}_t)_t$ to P_j

Data recovery:
Upon receiving $(\mathsf{hp}_t, \mathrm{M}_t)_{t=1,\ldots,k}$, P_j computes $\mathrm{K}_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, \mathfrak{L}_s, (\ell, \mathrm{C}), \delta)$ and gets $m_s \leftarrow \mathrm{R} \oplus \mathrm{K}_s \oplus \mathrm{M}_s$.
Then P_j erases everything except m_s and s.

Idea of the Construction

In [**ABB$^+$13**], we give a way to construct a UC-secure oblivious transfer protocol from an SPHF-friendly commitment, but we only give an instantiation of such an SPHF-friendly commitment in a DDH-based setting, using Haralambiev commitment scheme [Har11] and Cramer-Shoup encryption scheme [CS02].

As shown in Section 4.3.3, we show in [**BC15**] how to construct, in a generic way, a UC-secure SPHF-friendly commitment scheme in any setting, from a collision-resistant chameleon hash and a CCA-2 encryption scheme. This enables us to strengthen the generic part of the [**ABB$^+$13**] construction, by showing how to construct a UC-secure oblivious transfer from any collision-resistant chameleon hash and CCA-2 encryption scheme.

In the protocol described in [**ABB$^+$13**], from a high point of view[1], the user sends to the server a commitment of the number s of the line it is willing to obtain. The server then computes a pair of keys for a smooth projective hash function (SPHF) adapted to the commitment. It keeps secret the hash key and sends the projection key to the user, along with the hash value of all the lines of the database. Thanks to the properties of the SPHF, the user will then be able to recover the particular line it wants, using the public projection key and the secret random coins it used to create its committed value in the first place. The properties of the SPHF also ensure that the server has no idea about the line the user is requiring, and that the user cannot obtain any information from the hash values of the other lines of DB, which are exactly the requirements of a secure OT.

We prove the security of this protocol in the UC framework, which implies the use of a commitment with strong security properties. Indeed, the simulator of a user needs to be able to change its mind about the line required, hence an *equivocable* commitment; and the simulator of a server also needs to be able to extract the line required by the user, hence an *extractable* commitment. Unfortunately, as explained in Section 4.3.1, combining both equivocability and extractability on the same commitment scheme, especially if we require this commitment scheme to admit an SPHF, is a difficult task and requires more security properties, namely it to be SPHF-friendly.

[1]Note that we omit here for the sake of simplicity the creation of a secure channel between the user and the server (this is only needed in the adaptive version of the protocol).

Generic Protocol

We denote by DB the database of the server containing $k = 2^d$ lines, and s the line requested by the user in an oblivious way. We assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and a $\mathsf{IND\text{-}CPA}$ encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. We also assume the existence of compatible CCA-encryption and chameleon hash with the properties described in [**BC15**], and we generically obtain from them the SPHF-friendly commitment scheme given in the paper.

We exactly follow the construction given in [**ABB$^+$13**], giving the protocol presented on Figure 6.4. The only difference is that we take advantage of the pre-flow to ask the server to generate the chameleon hash verification keys $(\mathsf{vk}, \mathsf{vtk})$. For the sake of simplicity, we only give the version for adaptive security, in which the server generates pk and c to create a somewhat secure channel (they would not be used in the static version).

Fig. 6.4 – Adaptive UC-Secure 1-out-of-k OT from an SPHF-Friendly Commitment [ABB$^+$13]

CRS generation:
$\rho = (\mathsf{ek}, \mathsf{ck}, \mathsf{param}) \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow:
1. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen
2. Server generates a verification key pair $(\mathsf{vk}, \mathsf{vtk}) \xleftarrow{\$} \mathsf{VKeyGen}(\mathsf{ck})$ for CH, stores vtk and completely erases the random coins used by $\mathsf{VKeyGen}$
3. Server sends pk and vk to User

Index query on s:
1. User chooses a random value J, computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, \mathrm{J})$
2. User computes $(\mathrm{C}, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$
3. User stores δ and completely erases J, R and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Server

Database input $(m_1, \ldots, m_k)$:
1. Server decrypts $\mathrm{J} \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and then $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$
2. For $t = 1, \ldots, k$: Server computes $\mathsf{hk}_t \xleftarrow{\$} \mathsf{HashKG}(\mathfrak{L}_t, \mathsf{param})$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, (\mathfrak{L}_t, \mathsf{param}), (\ell, \mathrm{C}))$, $\mathrm{K}_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, (\mathfrak{L}_t, \mathsf{param}), (\ell, \mathrm{C}))$, and $\mathrm{M}_t \leftarrow \mathrm{R} \oplus \mathrm{K}_t \oplus m_t$
3. Server erases everything except $(\mathsf{hp}_t, \mathrm{M}_t)_{t=1,\ldots,k}$ and sends them over a secure channel

Data recovery:
Upon receiving $(\mathsf{hp}_t, \mathrm{M}_t)_{t=1,\ldots,k}$, user computes $\mathrm{K}_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathfrak{L}_s, \mathsf{param}), (\ell, \mathrm{C}), \delta)$ and gets $m_s \leftarrow \mathrm{R} \oplus \mathrm{K}_s \oplus \mathrm{M}_s$.

The oblivious transfer scheme described in Figure 6.4 is UC-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels, as soon as the commitment scheme is constructed from a secure publicly-verifiable chameleon hash and a secure CCA encryption scheme admitting an SPHF on the language of valid ciphertexts.

Instanciations

Instanciation Based on Cramer-Shoup Encryption (DDH). We give in the paper a construction based on DDH. The commitment revisits the one used in [**ABB$^+$13**] but we remove the pairing used in it thanks to the methods described in the paper, by generating vtk on the fly. For the chameleon hash, we use a CDH-based Pedersen encryption scheme. However, as such CH is not designated verifier, we transform it in an Haralambiev way [Har11, Section 4.1.4]. For the CCA encryption we rely on an extended version of Cramer-Shoup encryption.

Instanciation Based on Paillier Encryption (Composite Residuosity). The solution is pretty straightforward on how to instantiate the previous scheme while relying on a DCR assumption. This simply requires the generic transformation from any native DDH scheme into a DCR based one presented in [HO09].

It is interesting to note that this boils down to using the Paillier-based CCA encryption presented in [CS02], in addition to a DCR-based Chameleon Hash encryption.

Instanciation Based on Dual Regev Encryption (LWE). We present in the paper a Chameleon Hash constructed from the SIS assumption, following the chameleon hash given in [CHKP10] but using the Micciancio-Peikert trapdoor generation [MP12].

Katz and Vaikuntanathan proposed in [KV09] a labelled CCA-Encryption with an approximate SPHF. In order to achieve the $2d$-labelled, one simply has to use the same label in all the encryptions, and then add a one-time signature, built for example by using the previous chameleon hash.

The approximate SPHF presented in [KV09] is sufficient for our application with a small modification to our generic framework. Indeed, instead of obtaining two identical values for Hash and ProjHash, the correctness only guarantees that for a well-formed ciphertext, those two values have a small Hamming distance, hence xoring the two values together leads to a string with low Hamming weight. Assuming the line in the database is first encoded using an Error Correcting Code, and then masked by the server using the Hash value, the user can then use his projective hash value to recover a word near a valid encoding for the required entry, and then decoding using the Error Correcting Code as the remaining noise is small, he will recover the valid string. On invalid lines, the noise is seemingly random, hence beyond the decoding limit of any possible code.

6.2.3 Improving the Complexity

As already seen in Section 5.2.2 for PAKE schemes, we show in [**BC16**] that the UC-commitment from [FLM11] (while not fitting with the methodology of traditional SPHF from [**ABB$^+$13**]), is compatible with SP-SPHF and can be used to build UC protocols.

We provide a construction of a three-round UC-secure 1-out-of-k OT with adaptive security based on MDDH (page 262). Assuming reliable erasures and a single global CRS, we show that our instantiation is UC-secure against adaptive adversaries. Besides having a lesser number of rounds than most recent existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solutions so far (see Figure 6.10 page 74).

We denote by DB the database of the server containing $k = 2^d$ lines, and s the line requested by the user in an oblivious way. We assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and a IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. The commitment used is the variant of [FLM11] described above. It is denoted as Com^{ℓ} in the description of the scheme, with ℓ being a label. Note that sid denotes the session identifier, ssid the subsession identifier and cid the commitment identifier and that the combination (sid, cid) is globally unique, as in [HMQ04, FLM11].

We present our construction in Figure 6.5, following the global framework presented in [**ABB$^+$13**], for an easier efficiency comparison. The proof of the following result can be found in the paper, page 274.

Theorem 5 ([**BC16**]). *The oblivious transfer scheme described in Figure 6.5 is UC-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

6.3 Adaptive version of the Protocol

6.3.1 Definition and (New) Security Model

The classical OT constructions based on the commitment/SPHF paradigm (with so-called implicit decommitment), among the latest in the UC framework [CKWZ13], [**ABB$^+$13, BC15**], require the server

Fig. 6.5 – Adaptive UC-Secure 1-out-of-k OT from an SPHF-Friendly Commitment [BC16]

CRS generation:
$\mathsf{crs} \stackrel{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \stackrel{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow:
1. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen
2. Server sends pk to User

Index query on s:
1. User chooses a random value J, computes $\mathrm{S} \leftarrow \mathrm{F}(\mathrm{J})$ and encrypts J under pk:
   $c \stackrel{\$}{\leftarrow} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, \mathrm{J})$
2. User computes $([\vec{\mathrm{C}}]_1, [\vec{\mathrm{R}}]_2, [\vec{\Pi}]_1) \stackrel{\$}{\leftarrow} \mathsf{Com}^{\ell}(\mathsf{crs}, s, \mathsf{sid}, \mathsf{cid}, \mathrm{P}_i, \mathrm{P}_j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$
3. User stores $[\vec{\Pi}]_1$ and completely erases J and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $[\vec{\mathrm{C}}]_1, [\vec{\mathrm{R}}]_2$ and c to Server

Database input $(m_1, \ldots, m_k)$:
1. Server decrypts $\mathrm{J} \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $\mathrm{S} \leftarrow \mathrm{F}(\mathrm{J})$
2. For $t = 1, \ldots, k$: Server computes $\mathsf{hk}_t \stackrel{\$}{\leftarrow} \mathsf{HashKG}(\mathfrak{L}_t)$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, \mathfrak{L}_t)$,
   $\mathrm{K}_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, (\mathfrak{L}_t, (\ell, [\vec{\mathrm{C}}]_1, [\vec{\mathrm{R}}]_2)))$ and $\mathrm{M}_t \leftarrow \mathrm{S} \oplus \mathrm{K}_t \oplus m_t$
3. Server erases everything except $(\mathsf{hp}_t, \mathrm{M}_t)_{t=1,\ldots,k}$ and sends them over a secure channel

Data recovery:
Upon receiving $(\mathsf{hp}_t, \mathrm{M}_t)_{t=1,\ldots,k}$, User computes $\mathrm{K}_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathfrak{L}_s, \ell, [\vec{\mathrm{C}}]_1, [\vec{\mathrm{R}}]_2), [\vec{\Pi}]_1)$ and gets $m_s \leftarrow \mathrm{S} \oplus \mathrm{K}_s \oplus \mathrm{M}_s$.

to send an encryption of the complete database for each line required by the user (thus $\mathrm{O}(k)$ each time). We give in [**BCG16**] a protocol (page 215) requiring $\mathrm{O}(\log(k))$ for each line (except the first one, still in $\mathrm{O}(k)$), in the UC framework with adaptive corruptions under classical assumptions (MDDH). This protocol builds upon the one we give in [**BC15**], which is the more efficient known scheme secure in the UC framework, and we use ideas from [GH07] to make it adaptive.

Using implicit decommitment in the UC framework implies a very strong commitment primitive (formalized as SPHF-friendly commitments in [**ABB$^+$13**]), which is both extractable and equivocable. Our idea is here to split these two properties by using on the one hand an equivocable commitment and on the other hand an (extractable) CCA encryption scheme by generalizing the way to access a line in the database. But this is infeasible with simple line numbers. Indeed, we suggest here not to consider anymore the line numbers as numbers in $\{1, \ldots, k\}$ but rather to “encode” them (the exact encoding will depend on the protocol): For every line t, a word W_t in the language $\mathfrak{L}_t$ will correspond to a representation of line t. This representation must be publicly verifiable, in the sense that anyone can associate t to a word W_t. We formalize this in the following definition of oblivious transfer[2], given without loss of generality[3] (the classical notion of OT being easily captured using $\mathfrak{L}_t = \{t\}$).

Languages

The language $\mathfrak{L} \subset \mathrm{X}$ used in the definition of an SPHF should be a hard-partitioned subset of X, i.e. it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element not in $\mathfrak{L}$ (see formal definition in [GL03, AP06]). The languages used here are more complex and should fulfill the following properties[4]:

[2]The adaptive version only implies that the database $(m_1, \ldots, m_k)$ is sent only once in the interaction, while the user can query several lines (i.e. several words), in an adaptive way.

[3]This formalization furthermore encompasses the variants of OT, such as conditioned OT, where a user accesses a line only if he knows a credential for this line.

[4]We here mainly consider languages which are hard-partitioned subsets, for instance, encryptions of publicly verifiable languages.

- *Publicly Verifiable:* Given a word x in X, anyone should be able to decide in polynomial time whether $x \in \mathfrak{L}$ or not.
- *Self-Randomizable*: Given a word in the language, anyone should be able to sample a new word in the language[5], and the distribution of this resampling should be indistinguishable from an honest distribution. This will be used in order to prevent an adversary, or the authority in charge of distributing the words, to learn which specific form of the word was used by the user.

In case we consider several languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$, we also assume it is a *Trapdoor Collection of Languages*: It is computationally hard to sample an element in $\mathfrak{L}_1 \cap \dots \cap \mathfrak{L}_n$, except if one possesses a trapdoor tk (without the knowledge of the potential secret keys)[6]. For instance, if for all i, $\mathfrak{L}_i$ is the language of the equivocable commitments on words in an inner language $\widetilde{\mathfrak{L}}_i = \{i\}$ (as we will consider for OT), the common trapdoor key can be the equivocation trapdoor.

Depending on the applications, we can assume a *Keyed Language*, which means that it is set by a trusted authority, and that it is hard to sample fresh elements from scratch in the language without the knowledge of a secret language key $\mathsf{sk}_{\mathfrak{L}}$. In this case, the authority is also in charge of giving a word in the language to the receiver.

In case the language is keyed, we assume it is also a *Trapdoor Language*: We assume the existence of a trapdoor tk_{L} allowing a simulator to sample an element in $\mathfrak{L}$ (without the knowledge of the potential secret key $\mathsf{sk}_{\mathfrak{L}}$). For instance, for a language of valid Waters signatures of a message M (as we will consider for OSBE), one can think of $\mathsf{sk}_{\mathfrak{L}}$ as being the signing key, whereas the trapdoor $\mathsf{tk}_{\mathfrak{L}}$ can be the discrete logarithm of h in basis g.[7]

Security Model

In an OT protocol, a server $\mathcal{S}$ possesses a database of k lines $(m_1, \dots, m_k) \in (\{0,1\}^{\mathfrak{K}})^k$. A user $\mathcal{U}$ will be able to recover m_s (in an oblivious way) as soon as he owns a word $\mathrm{W}_s \in \mathfrak{L}_s$. The languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$ will be assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Definition 3 (Oblivious Transfer)**.** *An OT scheme is defined by five algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DBGen}, \mathsf{Samp}, \mathsf{Verif})$*, along with an interactive protocol* $\mathsf{Protocol}\langle \mathcal{S}, \mathcal{U} \rangle$*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the number* k*;*

or $\mathsf{SetupT}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[8] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$*.*

- $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ *generates, for all* $t \in \{1, \dots, k\}$*, the description of the language* $\mathfrak{L}_t$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_t}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$*.*
- $\mathsf{Samp}(\mathsf{param})$ *or* $\mathsf{Samp}(\mathsf{param}, (\mathsf{sk}_{\mathfrak{L}_t})_{t \in \{1,\dots,k\}})$ *generates a word* $\mathrm{W}_t \in \mathfrak{L}_t$*;*
- $\mathsf{Verif}_t(\mathrm{W}_t, \mathfrak{L}_t)$ *checks whether* W_t *is a valid word in the language* $\mathfrak{L}_t$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*
- $\mathsf{Protocol}\langle \mathcal{S}((\mathfrak{L}_1, \dots, \mathfrak{L}_k), (m_1, \dots, m_k)), \mathcal{U}((\mathfrak{L}_1, \dots, \mathfrak{L}_k), \mathrm{W}_s) \rangle$ *between the server* $\mathcal{S}$ *with the private database* $(m_1, \dots, m_k)$ *and corresponding languages* $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$*, and the user* $\mathcal{U}$ *with the same languages and the word* W_s*, proceeds as follows. If the algorithm* $\mathsf{Verif}_s(\mathrm{W}_s, \mathfrak{L}_s)$ *returns* 1*, then* $\mathcal{U}$ *receives* m_s*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

The ideal functionality of an Oblivious Transfer (OT) protocol recalled in Figure 6.1 was given in [Can01, CKWZ13], [**ABB+13**], and an adaptive version in [GH08]. We here combine them and rewrite it in simple UC and using our language formalism (instead of directly giving a number line s to

[5]It should be noted that this property is not incompatible with the potential secret key of the language in case it is keyed (see below).

[6]This implicitly means that the languages are compatible, in the sense that one can indeed find a word belonging to all of them.

[7]As another example, one may think of more expressive languages which may not rely directly on generators fixed by the CRS. In this case, one can assume that the CRS contains parameters for an encryption and an associated NIZK proof system. The description of such a language is thus supplemented with an encryption of the language trapdoor, and a non-interactive zero-knowledge proof that the encrypted value is indeed a trapdoor for the said language. Using the knowledge of the decryption key, the simulator is able to recover the trapdoor.

[8]The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

the functionality, the user will give it a word $W_s \in \mathfrak{L}_s$). The resulting functionality $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$ is given in Figure 6.6. Recall that there is no need to give an explicit description of the corruptions in the simple version of UC [CCL15].

Fig. 6.6 – Ideal Functionality for (Adaptive) Oblivious Transfer $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$

The functionality $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_k)$ along with the corresponding public verification algorithms $(\mathsf{Verif}_1, \ldots, \mathsf{Verif}_k)$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1, \ldots, P_n$ via the following queries:

- **Upon receiving an input $(\mathsf{NewDataBase}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ from party P_i,** with $m_t \in \{0,1\}^{\mathfrak{K}}$ for all t: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further NewDataBase-message with the same ssid from P_i.
- **Upon receiving an input $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, W_s)$ from party P_j:** ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ is not recorded. Otherwise, reveal $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$ and send $(\mathsf{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, m'_s)$ to P_j where $m'_s = m_s$ if $\mathsf{Verif}_s(W_s, \mathfrak{L}_s)$ returns 1, and $m'_s = \perp$ otherwise.

  (Non-Adaptive case: Ignore further Receive-message with the same ssid *from* P_j*.)*

6.3.2 High-Level Idea of the Construction

Our construction in [**BCG16**] (pages 216 to 221) builds upon the UC-secure OT scheme from [**BC15**], with ideas inspired from [GH07], who propose a neat framework allowing to achieve *adaptive* Oblivious Transfer (but not in the UC framework). Their construction is quite simple: It requires a *blind Identity-Based Encryption*, in other words, an IBE scheme in which there is a way to query for a user key generation without the authority (here the server) learning the targeted identity (here the line in the database). Once such a Blind IBE is defined, one can conveniently obtain an oblivious transfer protocol by asking the database to encrypt (once and for all) each line for an identity (the t-th line being encrypted for the identity t), and having the user do a blind user key generation query for identity s in order to recover the key corresponding to the line s he expects to learn.

This approach is round-optimal: After the database preparation, the first flow is sent by the user as a commitment to the identity s, and the second one is sent by the server with the blinded expected information. But several technicalities arise because of the UC framework we consider here. For instance, the blinded expected information has to be masked, we do this here thanks to an SPHF. Furthermore, instead of using simple line numbers as identities, we have to commit to words in specific languages (so as to ensure extractability and equivocability) as well as to *fragment* the IBE keys into bits in order to achieve $O(\log k)$ in both flows. This allows us to achieve the first UC-secure adaptive OT protocol allowing adaptive corruptions.

Following [BKP14], we recall in the paper the definitions, notations and security properties for an IBE scheme, seen as an Identity-Based Key Encapsulation (IBKEM) scheme. We continue to follow the KEM formalism by adapting the definition of a Blind IBE scheme given in [GH07] to this setting.

Definition 4 (Blind Identity-Based Key Encapsulation Scheme)**.** *A Blind Identity-Based Key Encapsulation scheme BlindIBKEM consists of four PPT algorithms (Gen, BlindUSKGen, Enc, Dec) with the following properties:*

- *Gen, Enc and Dec are defined as for a traditional IBKEM scheme.*
- *$\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ is an interactive protocol, in which an honest user $\mathcal{U}$ with identity $\mathsf{id} \in \mathcal{ID}$ obtains the corresponding user secret key $\mathsf{usk}[\mathsf{id}]$ from the master authority $\mathcal{S}$ or outputs an error message, while $\mathcal{S}$'s output is nothing or an error message (ℓ is a label and $\boldsymbol{\rho}$ the randomness).*

Defining the security of a BlindIBKEM requires two additional properties, stated as follows (see [GH07, pages 6 and 7] for the formal security games):

1. **Leak-free Secret Key Generation** (called Leak-free Extract for Blind IBE security in the original paper): A potentially malicious user cannot learn anything by executing the BlindUSKGen protocol

with an honest authority which he could not have learned by executing the USKGen protocol with an honest authority; Moreover, as in USKGen, the user must know the identity for which he is extracting a key.

2. **Selective-failure Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol; Moreover, the authority cannot cause the BlindUSKGen protocol to fail in a manner dependent on the user's choice.

For our applications, we only need a weakened property for blindness:[9]

3. **Weak Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol.

We show in the paper how to obtain a BlindIBKEM scheme from any IBKEM scheme.

6.3.3 Generic Construction

We derive from here our generic construction of OT (depicted in Figure 6.7). A pairing-based instantiation is given in the paper (page 221). We additionally assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. First, the owner of the database generates the keys for such an IBE scheme, and encrypts each line t of the database for the identity t. Then when a user wants to request a given line, he runs the blind user key generation algorithm and recovers the key for the expected given line. This leads to the following security result, proven in the paper (page 236).

Theorem 6 ([**BCG16**]). *Assuming that BlindUSKGen is constructed as described above, the adaptive Oblivious Transfer protocol described in Figure 6.7 UC-realizes the functionality $\mathcal{F}_{OT}^{\mathfrak{L}}$ presented in Figure 6.6 with adaptive corruptions assuming reliable erasures.*

6.4 Extension to Oblivious Language-Based Envelope

The previous construction opens new efficient applications to the already known Oblivious-Transfer protocols. But what happens when someone wants some additional access control by requesting extra properties, like if the user is only allowed to ask two lines with the same parity bits, the user can only request lines for whose number has been signed by an authority, or even finer control provided through credentials? In [**BCG16**], we also propose to develop a new primitive, that we call Oblivious Language-Based Envelope (OLBE, see page 223). The idea generalizes that of Oblivious Transfer and OSBE (recalled in the paper) for p messages (with p polynomial in the security parameter $\mathfrak{K}$).

6.4.1 Definition of Oblivious Language-Based Envelope

In such a protocol, a sender $\mathcal{S}$ wants to send one or several private messages (up to $p \leqslant k$) among $(m_1, \dots, m_k) \in (\{0,1\}^{\mathfrak{K}})^k$ to a recipient $\mathcal{R}$ in possession of a word $\mathrm{W} = (\mathrm{W}_{t_1}, \dots, \mathrm{W}_{t_p})$ such that some of the words W_{t_s} may belong to the corresponding language $\mathfrak{L}_{t_s}$. More precisely, the receiver gets each m_{t_s} as soon as $\mathrm{W}_{t_s} \in \mathfrak{L}_{t_s}$ with the requirement that he gets at most p messages. In such a scheme, the languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$ are assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable.

The collections of words can be a single certificate/signature on a message M (encompassing OSBE, with $k = p = 1$), a password, a credential, a line number (encompassing 1-out-of-k oblivious transfer[10], with $p = 1$), q line numbers (encompassing q-out-of-k oblivious transfer, with $p = q$), etc. (see the paper for detailed examples). Following the definitions for OSBE recalled in the paper and given in [LDB03, BPV12], we give the following definition for OLBE. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

[9]Two things to note: First, Selective Failure would be considered as a Denial of Service in the Oblivious Transfer setting. Then, we do not restrict ourselves to schemes where the blindness adversary has access to the generated user keys, as reliable erasures in the OT protocol provide us a way to forget them before being corrupted (otherwise we would need to use a randomizable base IBE).

[10]Even if, as explained in the former section, we would rather consider equivocable commitments of line numbers than directly line numbers, in order to get adaptive UC security.

Fig. 6.7 – Adaptive UC-Secure 1-out-of-k OT from a Fragmented Blind IBE

CRS generation:
$\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Database Preparation:
1. Server runs $\mathsf{Gen}(\mathfrak{K})$, to obtain $\mathsf{mpk}, \mathsf{msk}$.
2. For each line t, he computes $(\mathrm{D}_t, \mathrm{K}_t) = \mathsf{Enc}(\mathsf{mpk}, t)$, and $\mathrm{L}_t = \mathrm{K}_t \oplus \mathrm{DB}(t)$.
3. He also computes $\mathsf{usk}[t, b]$ for all $t = 1 \ldots, k$ and $b = 0, 1$ and erases msk.
4. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
5. He then publishes $\mathsf{mpk}, \{(\mathrm{D}_t, \mathrm{L}_t)\}_t, \mathsf{pk}$.

Index query on s:
1. User chooses a random value S, computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{S})$ and encrypts S under pk:
   $c \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, \mathrm{S})$
2. User computes $\mathcal{C}$ with the first flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{U}, \mathcal{S})$.
3. User stores the random $\boldsymbol{\rho}_s = \{\boldsymbol{\rho}_*\}$ needed to open $\mathcal{C}$ to s, and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $(c, \mathcal{C})$ to the Server

IBE input msk:
1. Server decrypts $\mathrm{S} \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{S})$
2. Server runs the second flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ on $\mathcal{C}$.
3. Server erases every new value except $(\mathsf{hp}_{t,b})_{t,b}, (\mathsf{busk}[t, b])_{t,b}, \mathrm{Z} \oplus \mathrm{R}$ and sends them over a secure channel.

Data recovery:
1. Uusing $\boldsymbol{\rho}_s$, user then recovers $\mathsf{usk}[s]$ from the values received from the server.
2. He then recovers the expected information with $\mathsf{Dec}(\mathsf{usk}[s], s, \mathrm{D}_s) \oplus \mathrm{L}_s$ and erases everything else.

Definition 5 (Oblivious Language-Based Envelope)**.** *An* OLBE *scheme is defined by four algorithms* (Setup, KeyGen, Samp, Verif)*, and one interactive protocol* Protocol$\langle\mathcal{S}, \mathcal{R}\rangle$*:*

- $\mathsf{Setup}(1^{\mathfrak{K}})$*, where $\mathfrak{K}$ is the security parameter, generates the global parameters* param*, among which the numbers k and p;*

or $\mathsf{SetupT}(1^{\mathfrak{K}})$*, where $\mathfrak{K}$ is the security parameter, additionally allows the existence*[11] *of a trapdoor* tk *for the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_k)$.*
- $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ *generates, for all $t \in \{1, \ldots, k\}$, the description of the language $\mathfrak{L}_t$ (as well as the language key $\mathsf{sk}_{\mathfrak{L}_t}$ if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_k)$.*
- $\mathit{Samp}(\mathsf{param}, \mathrm{I})$ *or* $\mathit{Samp}(\mathsf{param}, \mathrm{I}, (\mathsf{sk}_{\mathfrak{L}_s})_{s\in\mathrm{I}})$ *such that $\mathrm{I} \subset \{1, \ldots, k\}$ and $|\mathrm{I}| = p$, generates a list of words $(\mathrm{W}_s)_{s\in\mathrm{I}}$ such that $\mathrm{W}_s \in \mathfrak{L}_s$ for all $s \in \mathrm{I}$;*
- $\mathsf{Verif}_s(\mathrm{W}_s, \mathfrak{L}_s)$ *checks whether W_s is a valid word in the language $\mathfrak{L}_s$. It outputs 1 if the word is valid, 0 otherwise;*
- $\mathit{Protocol}\langle\mathcal{S}((\mathfrak{L}_1, \ldots, \mathfrak{L}_k), (m_1, \ldots, m_k)), \mathcal{R}((\mathfrak{L}_1, \ldots, \mathfrak{L}_k), (\mathrm{W}_s)_{s\in\mathrm{I}})\rangle$ *between the sender $\mathcal{S}$ with private messages $(m_1, \ldots, m_k)$ and corresponding languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$, and the recipient $\mathcal{R}$ with the same languages and the words $(\mathrm{W}_s)_{s\in\mathrm{I}}$ with $\mathrm{I} \subset \{1, \ldots, k\}$ and $|\mathrm{I}| = p$, proceeds as follows. For all $s \in \mathrm{I}$, if the algorithm* $\mathsf{Verif}_s(\mathrm{W}_s, \mathfrak{L}_s)$ *returns 1, then $\mathcal{R}$ receives m_s, otherwise it does not. In any case, $\mathcal{S}$ does not learn anything.*

6.4.2 Security Properties and Ideal Functionality of OLBE

Since we aim at proving the security in the universal composability framework, we now describe the corresponding ideal functionality (depicted in Figure 6.8). However, in order to ease the comparison with an OSBE scheme, we first list the security properties required, following [LDB03] and [BPV12]:

[11]The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

- *correct*: the protocol actually allows $\mathcal{R}$ to learn $(m_s)_{s\in\mathrm{I}}$, whenever $(\mathrm{W}_s)_{s\in\mathrm{I}}$ are valid words of the languages $(\mathfrak{L}_s)_{s\in\mathrm{I}}$, where $\mathrm{I} \subset \{1,\dots,k\}$ and $|\mathrm{I}| = p$;
- *semantically secure* (*sem*): the recipient learns nothing about the input m_s of $\mathcal{S}$ if it does not use a word in $\mathfrak{L}_s$. More precisely, if $\mathcal{S}_0$ owns $m_{s,0}$ and $\mathcal{S}_1$ owns $m_{s,1}$, the recipient that does not use a word in $\mathfrak{L}_s$ cannot distinguish between an interaction with $\mathcal{S}_0$ and an interaction with $\mathcal{S}_1$ even if the receiver has seen several interactions $\langle \mathcal{S}((\mathfrak{L}_1,\dots,\mathfrak{L}_k),(m_1,\dots,m_k)), \mathcal{R}((\mathfrak{L}_1,\dots,\mathfrak{L}_k),(\mathrm{W}'_t)_{t\in\mathrm{I}})\rangle$ with valid words $\mathrm{W}'_s \in \mathfrak{L}_s$, and the same sender's input m_s;
- *escrow free (oblivious with respect to the authority)*: the authority corresponding to the language $\mathfrak{L}_s$ (owner of the language secret key $\mathsf{sk}_{\mathfrak{L}_s}$ – if it exists), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a word W_s in the language $\mathfrak{L}_s$ or not. This requirement also holds for anyone holding the trapdoor key tk.
- *semantically secure w.r.t. the authority* (*sem**): after the interaction, the trusted authority (owner of the language secret keys if they exist) learns nothing about the values $(m_s)_{s\in\mathrm{I}}$ from the transcript of the execution. This requirement also holds for anyone holding the trapdoor key tk.

Moreover, the Setups should be indistinguishable and it should be infeasible to find a word belonging to two or more languages without the knowledge of tk.

Fig. 6.8 – Ideal Functionality for Oblivious Language-Based Envelope $\mathcal{F}_{\mathbf{OLBE}}$

The functionality $\mathcal{F}_{\mathrm{OLBE}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1,\dots,\mathfrak{L}_k)$ along with the corresponding public verification algorithms $(\mathsf{Verif}_1,\dots,\mathsf{Verif}_k)$. It interacts with an adversary $\mathcal{S}$ and a set of parties $\mathrm{P}_1,\dots,\mathrm{P}_n$ via the following queries:

- **Upon receiving an input $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1,\dots,m_k))$ from party P_i**, with $m_t \in \{0,1\}^{\mathfrak{K}}$ for all t: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1,\dots,m_k))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.

- **Upon receiving an input $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (\mathrm{W}_s)_{s\in\mathrm{I}})$ where $\mathrm{I} \subset \{1,\dots,k\}$ and $|\mathrm{I}| = p$ from party P_j**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m_1,\dots,m_k))$ is not recorded. Otherwise, reveal $(\mathsf{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j)$ to the adversary $\mathcal{S}$ and send $(\mathsf{Received}, \mathsf{sid}, \mathsf{ssid}, \mathrm{P}_i, \mathrm{P}_j, (m'_s)_{s\in\mathrm{I}})$ to P_j where $m'_s = m_s$ if $\mathsf{Verif}_s(\mathrm{W}_s, \mathfrak{L}_s)$ returns 1, and $m'_s = \perp$ otherwise. Ignore further $\mathsf{Received}$-message with the same ssid from P_j.

The ideal functionality is parametrized by a set of languages $(\mathfrak{L}_1,\dots,\mathfrak{L}_k)$. Since we show in the paper that one can see OSBE and OT as special cases of OLBE, it is inspired from the oblivious transfer functionality given in [Can01, CKWZ13], [**ABB+13**] in order to provide a framework consistent with works well-known in the literature. As for oblivious transfer (Figure 6.6), we adapt them to the simple UC framework for simplicity (this enables us to get rid of Sent and Received queries from the adversary since the delayed outputs are automatically considered in this simpler framework: We implicitly let the adversary determine if it wants to acknowledge the fact that a message was indeed sent). The first step for the sender (Send query) consists in telling the functionality he is willing to take part in the protocol, giving as input his intended receiver and the messages he is willing to send (up to p messages). For the receiver, the first step (Receive query) consists in giving the functionality the name of the player he intends to receive the messages from, as well as his words. If the word does belong to the language, the receiver recovers the sent message, otherwise, he only gets a special symbol $\perp$.

6.4.3 Generic UC-Secure Instantiation of OLBE with Adaptive Security

For the sake of clarity, we now concentrate on the specific case where $p = 1$. This is the most classical case in practice, and suffices for both OSBE and 1-out-of-k OT. In order to get a generic protocol in which $p > 1$, one simply has to run p protocols in parallel. This modifies the algorithms Samp and Verify as follows: $\mathsf{Samp}(\mathsf{param}, \{s\})$ or $\mathsf{Samp}(\mathsf{param}, \{s\}, \{\mathsf{sk}_{\mathfrak{L}_s}\})$ generates a word $\mathrm{W} = \mathrm{W}_s \in \mathfrak{L}_s$ and $\mathsf{Verif}_t(\mathrm{W}, \mathfrak{L}_t)$ checks whether W is a valid word in $\mathfrak{L}_t$.

Let us introduce our protocol OLBE: we will call $\mathcal{R}$ the receiver and $\mathcal{S}$ the sender. If $\mathcal{R}$ is an honest receiver, then he knows a word $\mathrm{W} = \mathrm{W}_s$ in the language $\mathfrak{L}_s$. If $\mathcal{S}$ is an honest sender, then he wants to send up a message among $(m_1,\dots,m_k) \in (\{0,1\}^{\mathfrak{K}})^k$ to $\mathcal{R}$. We assume the languages $\mathfrak{L}_t$ to be self-randomizable and publicly verifiable. We also assume the collection of languages $(\mathfrak{L}_1,\dots,\mathfrak{L}_k)$ possess a trapdoor, that the simulator is able to find by programming the common reference string. As recalled

Fig. 6.9 – UC-Secure OLBE for One Message (Secure Against Adaptive Corruptions)

CRS generation:
$\mathsf{param} \overset{\$}{\leftarrow} \mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cca}} \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cca}}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cpa}} \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow:
1. Sender generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
2. Sender sends pk to User.

Flow From the Receiver $\mathcal{R}$:
1. User chooses a random value J, computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$ and encrypts J under pk:
   $c \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, \mathrm{J})$.
2. User computes $\mathrm{C} \overset{\$}{\leftarrow} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathrm{W}; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$.
3. User completely erases J and the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Sender. He also checks the validity of his words: the receiver only keeps the random coins used by $\mathsf{Encrypt}_{\mathrm{cca}}$ for the t such that $\mathsf{Verif}_t(\mathrm{W}, \mathfrak{L}_t) = 1$ (since he knows they will be useless otherwise).

Flow From the Sender $\mathcal{S}$:
1. Sender decrypts $\mathrm{J} \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and then $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$.
2. For all $t \in \{1, \dots, k\}$, sender computes $\mathsf{hk}_t = \mathsf{HashKG}(\ell, \mathfrak{L}_{\mathrm{C},t}, \mathsf{param})$,
   $\mathsf{hp}_t = \mathsf{ProjKG}(\mathsf{hk}_t, \ell, (\mathfrak{L}_{\mathrm{C},t}, \mathsf{param}))$, $v_t = \mathsf{Hash}(\mathsf{hk}_t, (\mathfrak{L}_{\mathrm{C},t}, \mathsf{param}), (\ell, \mathrm{C}))$, $\mathrm{Q}_t = m_t \oplus \mathtt{KDF}(v_t) \oplus \mathrm{R}$.
3. Sender erases everything except $(\mathrm{Q}_t, \mathsf{hp}_t)_{t \in \{1,\dots,k\}}$ and sends them over a secure channel.

Message recovery:
Upon receiving $(\mathrm{Q}_t, \mathsf{hp}_t)_{t \in \{1,\dots,k\}}$, $\mathcal{R}$ can recover m_s by computing
$m_s = \mathrm{Q}_s \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_{\mathrm{C},s}, \mathsf{param}), (\ell, \mathrm{C}), r) \oplus \mathrm{R}$.

previously, this trapdoor enables him to find a word lying in the intersection of the k languages. This should be infeasible without the knowledge of the trapdoor. Intuitively, this allows the simulator to commit to all languages at once, postponing the time when it needs to choose the exact language he wants to bind to. On the opposite, if a user was granted the same possibilities, this would prevent the simulator to extract the chosen language.

We assume the existence of a labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cca}}, \mathsf{KeyGen}_{\mathrm{cca}}, \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}, \mathsf{Decrypt}^{\ell}_{\mathrm{cca}})$ compatible with an SPHF onto a set G. In the KeyGen algorithm, the description of the languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$ thus implicitly defines the languages $(\mathfrak{L}_{\mathrm{C},1}, \dots, \mathfrak{L}_{\mathrm{C},k})$ of CCA-encryptions of elements of $(\mathfrak{L}_1, \dots, \mathfrak{L}_k)$. We additionally use a key derivation function $\mathtt{KDF}$ to derive a pseudo-random bit-string $\mathrm{K} \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in \mathrm{G}$. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma just lead to a security loss linear in the number of extractions. We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter.

We follow the ideas of the oblivious transfer constructions given in [**ABB$^+$13, BC15**], giving the protocol presented on Figure 6.9. For the sake of simplicity, we only give the version for adaptive security, in which the sender generates a public key pk and ciphertext c to create a somewhat secure channel (they would not be used in the static version). The proof of the following result is given in the paper, page 239.

Theorem 7 ([**BCG16**]). *The oblivious language-based envelope scheme described in Figure 6.9 is UC-secure in the presence of adaptive adversaries, assuming reliable erasures, an IND-CPA encryption scheme, and an IND-CCA encryption scheme admitting an SPHF on the language of valid ciphertexts of elements of $\mathfrak{L}_t$ for all t, as soon as the languages are self-randomizable, publicly-verifiable and admit a common trapdoor.*

6.4.4 Oblivious Primitives Obtained by the Framework

Classical oblivious primitives such as Oblivious Transfer (both 1-out-of-k and q-out-of-k) or Oblivious Signature-Based Envelope directly lie in this framework and can be seen as examples of Oblivious Language-Based Envelope. We provide in the paper details about how to describe the languages and choose appropriate smooth projective hash functions to readily achieve current instantiations of Oblivious Signature-Based Envelope or Oblivious Transfer from our generic protocol (pages 226 and 242). The framework also enables us to give a new instantiation of Access Controlled Oblivious Transfer under classical assumptions (page 245). In such a primitive, the user does not automatically gets the line he asks for, but has to prove that he possesses one of the credential needed to access this particular line.

For the sake of simplicity, all the instantiations given are pairing-based but techniques explained in [**BC15**] could be used to rely on other families of assumptions, like decisional quadratic residue or LWE.

6.5 From a UC-Secure PAKE to a UC-Secure OT

6.5.1 Introduction

In [**BCG17**], we propose a generic transformation from Password-Authenticated Key Exchange to Oblivious Transfer. As the reverse transformation was already studied in [CDVW12], this allows to study one protocol for the other. Our framework allows to transform a UC-secure PAKE into a UC-secure OT with the same level of security (for instance resistance to adaptive corruptions).

Our Construction

We choose to focus on the transformation of a 2-round PAKE to a 3-round OT since this kind of PAKE is the most commonly encountered and the most efficient one. Furthermore, this allows us to give a generic optimization of our transformation in this specific case, exploiting the fact that the server does not need to hide his “password”, since his password is a (public) number of line in the database. Note that our generic transformation could allow to transform an n_r-round PAKE into an $(n_r + 1)$-round OT, but the optimization would then have to be tailored to the considered protocol.

After showing an application of our technique on our PAKE scheme from [**ABB$^+$13**], which allows us to immediately recover the associated OT scheme given in our article, we show that our transformation also applies to the PAKE scheme from [JR15], allowing us to give a nearly optimal OT scheme.

Note that our technique works with every possible PAKE scheme, be they elliptic-curve-based or not. Furthermore, also note that it also allows to transform a BPR-secure PAKE [BPR00] into an OT secure in a game-based security model. In order to illustrate these last two points, we apply in the paper our framework to a (non-UC) secure lattice-based PAKE scheme proposed by [KV09], giving us a (non-UC) secure lattice-based OT scheme.

Comparison with other existing OT schemes

The table given in Figure 6.10 shows the costs of various known OT schemes based on elliptic curves.

While Barreto-Naehrig curves [BN06] are considered less and less secure for efficient parameters, we consider for the asymptotic scaling that elements in $\mathbb{G}_2$ are smaller than those in $\mathbb{G}_1$, hence, we switched some elements to $\mathbb{G}_2$ for big enough values of k.

Thanks to the use of a QA-NIZK technique, we manage to get rid of the extra logarithm overhead. And we end up being approximately four times more efficient than existing solutions without this overhead.

6.5.2 Generic Construction of a UC-Secure OT From a UC-Secure PAKE

There is a trend in proven cryptography that consists in finding the link between primitives and show which primitives can be used to build other ones. A well-known example is the transformation of an IBE

[12]It should be noted that our [**BC15**] OT does not rely on pairings. Classical implementations suggest that the elements in one of the groups in our scheme will be at least twice as big as those from the non-pairing curve, which would give roughly the same efficiency between both schemes in the 1-out-of-2 case, with an edge for the [**BC15**] construction in term of computational requirements. However, the scaling in this paper is then worse when increasing the number of lines (see the 1-out-of-k case).

Fig. 6.10 – Communication cost comparisons of various Elliptic Curve based OT schemes

Paper	Assumption	# Group elements	# Rounds
Static Security (1-out-of-2)			
[PVW08] + [GWZ09]	SXDH	51	8
[CKWZ13]	SXDH	$26 + 7\ \mathbb{Z}_p$	4
Adaptive Security (1-out-of-2)			
[ABB+13]	SXDH	$12\ \mathbb{G}_1 + 1\ \mathbb{G}_2 + 2\mathbb{Z}_p$	3
[BC15][12]	DDH	$15\ \mathbb{G} + 2\ \mathbb{Z}_p$	3
[BC16]	SXDH	$12\ \mathbb{G}_1 + 4\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
[BCG17]	SXDH	$6\ \mathbb{G}_1 + 2\ \mathbb{G}_2 + 2\ \mathbb{Z}_p$	3
Adaptive Security (1-out-of-k)			
[ABB+13]	SXDH	$\log k\ \mathbb{G}_1 + (k + 8 \log k)\ \mathbb{G}_2 + k\ \mathbb{Z}_p$	3
[BC15][12]	DDH	$(k + 9 \log k + 4)\ \mathbb{G} + 2k\ \mathbb{Z}_p$	3
[BC16]	SXDH	$4\ \mathbb{G}_1 + (4k + 4)\ \mathbb{G}_2 + k\ \mathbb{Z}_p$	3
[BCG17]	SXDH	$4\ \mathbb{G}_1 + (k + 2)\ \mathbb{G}_2 + k\ \mathbb{Z}_p$	3

encryption scheme into a signature scheme, which was proposed by Naor as recalled in [BF01], and its reverse transformation from an affine MAC or signature scheme into an IBE scheme given in [BKP14].

A neat result, presented by Canetti et al. in [CDVW12], shows how a UC-secure Oblivious Transfer scheme can be transformed into a UC-secure PAKE scheme. In this section we are going to go the other way, and show how a UC-secure PAKE can be transformed into a UC-secure Oblivious Transfer scheme. While considering this direction may seem like transforming a very strong primitive into a weaker one, this will allow us to prepare a framework to propose the most efficient Oblivious Transfer scheme to date. This also echoes to the results from [Ngu05] on a simulation-based transform.

Description of a UC-Secure PAKE Scheme

We consider PAKE protocols in two flows, since there already exists a multitude of PAKE schemes satisfying this requirement, and since minimizing the number of flows is one of the most important issues in modern instantiations. This allows us to generically give an optimization, leading to more efficient protocols. In order to present the framework or our transformation, we formally split a PAKE scheme into the following four algorithms[13].

- Setup: An algorithm that generates the initial crs.
- $\mathsf{flow}_1(\mathsf{pw}_j; \rho_j, \rho'_j)$: The algorithm run by the user P_j, using some randomness ρ_j and ρ'_j and possessing the password pw_j, to generate the first flow of the protocol. We note ρ_j the part of the randomness involved in hiding the password, and ρ'_j the remainder.
- $\mathsf{flow}_2(\mathsf{flow}_1, \mathsf{pw}_i; \rho_i, \rho'_i)$: The algorithm run by the user P_i, using some randomness ρ_i and possessing the password pw_i, after receiving the flow flow_1, to generate the second flow of the protocol. This algorithm also produces P_i's view of the key: $\mathrm{K}_j^{(i)}$.
- $\mathsf{Decap}(\mathsf{flow}_2, \mathsf{pw}_j, f(\rho_j, \rho'_j))$: The algorithm run by P_j when receiving the flow flow_2 from P_i, using his password pw_j and a function $f(\rho_j, \rho'_j)$ of the randomness initially used and the remainder (which function can be anything from the identity to the null function), to recover P_j's view of the key: $\mathrm{K}_i^{(j)}$.

Generic Construction of a UC-Secure OT Scheme

We denote by $\mathsf{DB} = (m_1, \ldots, m_k)$ the database of the server containing k lines, and by s the line requested by the user in an oblivious way. We assume the existence of a CPA-Secure encryption scheme, whose public parameters are going to be included in the public common reference string crs generated by the

[13]Since it was shown in [CHK+05] that UC-secure PAKE is impossible in the plain model, we focus on the CRS model for the ease of readability.

setup algorithm of the PAKE scheme. Such an encryption scheme $\mathcal{E}$ is described through four algorithms $(\mathsf{Setup}_{\rm cpa}, \mathsf{KeyGen}_{\rm cpa}, \mathsf{Encrypt}_{\rm cpa}, \mathsf{Decrypt}_{\rm cpa})$:

- $\mathsf{Setup}_{\rm cpa}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters $\mathsf{param}_{\rm cpa}$ of the scheme;
- $\mathsf{KeyGen}_{\rm cpa}(\mathsf{param}_{\rm cpa})$ outputs a pair of keys: a (public) encryption key pk and a (private) decryption key sk;
- $\mathsf{Encrypt}_{\rm cpa}(\mathsf{pk}, \mathrm{M}; \rho)$ outputs a ciphertext $\vec{c} = \mathcal{E}(\mathrm{M})$ on the message M, under the encryption key pk, using the randomness ρ;
- $\mathsf{Decrypt}_{\rm cpa}(\mathsf{sk}, \vec{c})$ outputs the plaintext M encrypted in the ciphertext $\vec{c}$, or $\perp$.

We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database.

Using the PAKE scheme, the encryption scheme and the PRG, one can now obtain an OT scheme from a PAKE scheme, as described in Figure 6.11. The idea of the protocol is as follows:

- First, a setup phase enables to generate the CRS, both for the PAKE and encryption schemes.
- The pre-flow is a technical requirement for the UC proof.
- When querying for line s, the receiver proceeds in three steps. The first one consists in generating a masking value R (technical requirement for the UC proof), the second one consists in running the first flow of the PAKE scheme for password s to generate the first flow denoted[14] as flow_0, and the third one consists in erasing anything but the needed values, and sending flow_0 to the sender.
- When answering to this query, the sender also proceeds in three steps. The first one consists in recovering the value R and the second one consists in running, for each line $t \in \{1, \dots, k\}$, the second flow of the PAKE scheme to generate the second flow denoted as flow_t (using t as the password) as well as $\mathcal{S}$'s view of the session key, denoted as $(\mathrm{K}_t)_{\mathcal{R}}^{(\mathcal{S})}$. Finally, in the third step, the server computes a xor Q_t of the line m_t and this session key and the value R and sends the set $(\mathsf{flow}_t, \mathrm{Q}_t)$ back to $\mathcal{R}$.
- When receiving this set of values, the receiver uses flow_s to run the decapsulation algorithm of the PAKE scheme (with password s) to recover his view of the session key, denoted as $(\mathrm{K}_s)_{\mathcal{S}}^{(\mathcal{R})}$. He finally uses R, Q_s and $(\mathrm{K}_s)_{\mathcal{S}}^{(\mathcal{R})}$ to recover the expected line m_s.

The correctness easily follows from the correctness of the PAKE protocol, since the views of $\mathcal{R}$ and $\mathcal{S}$ of the session key for the PAKE scheme executed for password s are the same. The scheme is oblivious with respect to $\mathcal{S}$ since the first flow of the PAKE scheme executed by $\mathcal{R}$ does not reveal any information on s. And it is oblivious with respect to $\mathcal{R}$ since the correctness of the PAKE scheme gives him no information on the session keys $(\mathrm{K}_t)_{\mathcal{R}}^{(\mathcal{S})}$ for $t \neq s$.

Theorem 8 ([**BCG17**]). *Under the UC-security of the PAKE protocol, the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database, and the IND-CPA security of the encryption scheme, the transformation presented in Figure 6.11 achieves a UC-secure 1-out-of-k Oblivious Transfer scheme, with the same handling of corruptions as in the PAKE protocol.*

Generic Optimization

It should be noted that in the previous transformation we never used the fact that the first user in the PAKE scheme (here, the receiver) does not learn the password from the second (here, the sender). This is a property required by PAKE but not useful in the transformation (since the sender executes k parallel PAKE schemes, using the (public) number t of the lines as the password). This argument is the key of the optimization we now propose in Figure 6.12. The difference is that in the index query, the receiver does not pick the second randomness ρ' and in the database answer, the randomness ρ_t is not used anymore. The proof remains the same as the previous one.

[14]Note that we now denote as flow_0 (and not flow_1) the flow generated by the PAKE algorithm flow_1, in order to avoid the confusion with the flow_1 message generated by the sender (for line 1) during the database answer phase.

Fig. 6.11 – UC-Secure 1-out-of-k OT from a UC-Secure PAKE

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{param}_{\text{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\text{cpa}}(1^{\mathfrak{K}})$.

Pre-Flow:
1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\text{cpa}}(\mathsf{param}_{\text{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\text{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:
1. $\mathcal{R}$ picks a random J, computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$ and sets $c \xleftarrow{\$} \mathsf{Encrypt}_{\text{cpa}}(\mathsf{pk}, \mathrm{J})$.
2. $\mathcal{R}$ picks a random (ρ, ρ'), and runs $\mathsf{flow}_1(s; \rho, \rho')$ to obtain flow_0.
3. $\mathcal{R}$ stores $f(\rho, \rho')$ needed for the Decap algorithm and R and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\text{cpa}}$ and sends (c, flow_0) to $\mathcal{S}$.

Database answer:
1. $\mathcal{S}$ decrypts $\mathrm{J} \leftarrow \mathsf{Decrypt}_{\text{cpa}}(\mathsf{sk}, c)$ and computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$.
2. For each line t:
   - $\mathcal{S}$ picks a random (ρ_t, ρ'_t), runs $\mathsf{flow}_2(\mathsf{flow}_0, t; \rho_t, \rho'_t)$ to generate flow_t and $(\mathrm{K}_t)^{(\mathcal{S})}_{\mathcal{R}}$.
   - $\mathcal{S}$ then computes $\mathrm{Q}_t = m_t \oplus (\mathrm{K}_t)^{(\mathcal{S})}_{\mathcal{R}} \oplus \mathrm{R}$.
3. $\mathcal{S}$ erases everything except $(\mathsf{flow}_t, \mathrm{Q}_t)_{t \in [\![1,k]\!]}$ and sends it to $\mathcal{R}$.

Data recovery:
$\mathcal{R}$ then using $f(\rho, \perp)$ computes $(\mathrm{K}_s)^{(\mathcal{R})}_{\mathcal{S}} = \mathsf{Decap}(\mathsf{flow}_s, s, f(\rho, \rho'))$, sets $m_s = \mathrm{Q}_s \oplus (\mathrm{K}_s)^{(\mathcal{R})}_{\mathcal{S}} \oplus \mathrm{R}$, and erases everything else.

Fig. 6.12 – Optimized UC-Secure 1-out-of-k OT from a UC-Secure PAKE

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{param}_{\text{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\text{cpa}}(1^{\mathfrak{K}})$.

Pre-Flow:
1. $\mathcal{S}$ generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\text{cpa}}(\mathsf{param}_{\text{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by $\mathsf{KeyGen}_{\text{cpa}}$.
2. $\mathcal{S}$ publishes pk.

Index query on s:
1. $\mathcal{R}$ picks a random J, computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$ and sets $c \xleftarrow{\$} \mathsf{Encrypt}_{\text{cpa}}(\mathsf{pk}, \mathrm{J})$.
2. $\mathcal{R}$ picks a random ρ, and runs $\mathsf{flow}_1(s; \rho, \perp)$ to obtain flow_0.
3. $\mathcal{R}$ stores $f(\rho, \perp)$ needed for the Decap algorithm and R and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\text{cpa}}$ and sends (c, flow_0) to $\mathcal{S}$.

Database answer:
1. $\mathcal{S}$ decrypts $\mathrm{J} \leftarrow \mathsf{Decrypt}_{\text{cpa}}(\mathsf{sk}, c)$ and computes $\mathrm{R} \leftarrow \mathrm{F}(\mathrm{J})$.
2. For each line t:
   - $\mathcal{S}$ picks a random ρ'_t, and runs $\mathsf{flow}_2(\mathsf{flow}_0, t; \perp, \rho'_t)$ to generate flow_t and $(\mathrm{K}_t)^{(\mathcal{S})}_{\mathcal{R}}$.
   - $\mathcal{S}$ then computes $\mathrm{Q}_t = m_t \oplus (\mathrm{K}_t)^{(\mathcal{S})}_{\mathcal{R}} \oplus \mathrm{R}$.
3. $\mathcal{S}$ erases everything except $(\mathsf{flow}_t, \mathrm{Q}_t)_{t \in [\![1,k]\!]}$ and sends it to $\mathcal{R}$.

Data recovery: $\mathcal{R}$ then using $f(\rho, \perp)$ computes $(\mathrm{K}_s)^{(\mathcal{R})}_{\mathcal{S}} = \mathsf{Decap}(\mathsf{flow}_s, s, f(\rho, \perp))$, sets $\mathrm{L}_s = \mathrm{Q}_s \oplus (\mathrm{K}_s)^{(\mathcal{R})}_{\mathcal{S}} \oplus \mathrm{R}$, and erases everything else.

Part IV

Conclusion

Conclusion and Perspectives

Working on concrete protocols proven in the UC framework enabled us in the several papers described in this manuscript to make improvements both on these protocols (in particular in terms of efficiency) and on the primitives used. We briefly recall here the main results obtained and give a few open problems that we plan to work on in the near future.

Smooth Projective Hash Functions. Following [CS02, GL03, CHK$^+$05], [**ACP09**], we focused on the now well-known paradigm combining commitments and smooth projective hash functions in order to construct UC-secure PAKE schemes. So far, only ad hoc SPHF had been constructed, and we proposed in our articles a full treatment of linear and quadratic pairing product equations.

However, Katz and Vaikuntanathan introduced in [KV11] a way to construct specific SPHF, which we called KV-SPHF (in which the projected hash key does not depend on the word in the language). It enabled them to propose the first one-round PAKE schemes, both in the BPR and the UC security models. We continued their work by proposing for instance the first KV-SPHF on the Cramer-Shoup encryption scheme, but this formalism only allows for linear equations so far. Maybe some applications would need more complex relations based on quadratic equations, and one could wonder how to construct the corresponding SPHF.

In all the SPHF designed so far, the witness (of the membership of the word to the language) usually was a scalar, when the word is a group element. Following work already done on signatures, we proposed the notion of structure-preserving SPHF, enabling to use a group element as a witness. This allowed us to get rid of the usual bit-per-bit trick and combine SPHF with more efficient commitments, leading to more efficient protocols.

Commitments. Considering the other classical building block of PAKE protocols, we first corrected and improved the best UC-secure commitment scheme known so far [Lin11a]. We then formalized precisely what security properties were needed in order for commitments to work well with SPHF, which is what we called SPHF-friendly commitments, and showed how to construct such commitments generically from chameleon hash and CCA encryption schemes. Finally, using our former notion of structure-preserving SPHF, we managed to obtain the same applications to PAKE and OT in a more efficient way, with a commitment [FLM11] which does not fulfill those requirements.

Password-Authenticated Key-Exchange. We continued the study of UC-secure PAKE started in [CHK$^+$05], [**ACP09**] by improving the complexity of the schemes. Furthermore, following the ideas of [KV11] and using the KV-SPHF mentionned above, we also managed to construct UC-secure one-round PAKE schemes fulfilling adaptive security (meaning that the adversary can corrupt a player anytime). We also used the same methodology to design more general schemes, denoted as LAKE (Language-Authenticated Key-Exchange), where the password is replaced by a word in a language.

Finally, we started the study of PAKE distributed over two or more servers in [**BCV16**], in a security model inspired from the BPR model [BPR00]. It would be worth designing an ideal functionality in the UC framework for such protocols, and to propose a scheme UC-secure in that model.

Oblivious Transfer. Although most of the work done on the primitives was initially designed for the construction of UC-secure PAKE schemes, we managed to apply it to OT schemes, giving us the most efficient UC-secure protocols so far. However, a drawback of using SPHF is to ask the server to send a very long message (of the size k of the database) for each query. We thus proposed a first step towards an adaptive version of these schemes, in which the server is only required to send a shorter message of length $\log(k)$. It is still an open problem to achieve a completely adaptive protocol, in which the server would be allowed to send a message of constant size.

Following the idea of OSBE (Oblivious Signature-Based Envelope), we also proposed the generic concept of OLBE (Oblivious Language-Based Envelope), in which a user obtains a message if and only if it owns the good credential for it (here a word in a language). This encompasses several other protocols, such as Access Controlled, Priced or Conditional Oblivious Transfer.

Finally, we continued the work started in [CDVW12], which shows how a UC-secure OT scheme can be transformed into a UC-secure PAKE scheme, by proving the transformation in the other direction, giving a close link between the two protocols studied in our work.

Other Security Models. We focused in all our work on the UC framework, which is one of the most powerful security model for multiparty protocols so far. But a recent trend in cryptography focuses on post-quantum security, which means security even in the case an adversary owns a quantum computer and can use it, for instance to break computational assumptions. It would be worth wondering whether our results still hold in such a setting, in particular by modifying the underlying assumptions to quantum-resistant ones.

List of Figures

Bibliography

[ABB$^+$13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, December 2013. (Cited on pages 7, 14, 16, 18, 19, 20, 32, 34, 36, 37, 39, 42, 43, 44, 49, 50, 61, 62, 63, 64, 65, 66, 67, 71, 72, 73, 74, and 81.)

[ABCP09] Michel Abdalla, Xavier Boyen, Céline Chevalier, and David Pointcheval. Distributed public-key cryptography from weak secrets. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 139–159. Springer, Heidelberg, March 2009. (Not cited.)

[ACCP08] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Efficient two-party password-based key exchange protocols in the UC framework. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 335–351. Springer, Heidelberg, April 2008. (Not cited.)

[ACCP09] Michel Abdalla, Dario Catalano, Céline Chevalier, and David Pointcheval. Password-authenticated group key agreement with adaptive security and contributiveness. In Bart Preneel, editor, *AFRICACRYPT 09*, volume 5580 of *LNCS*, pages 254–271. Springer, Heidelberg, June 2009. (Not cited.)

[ACFP05] Michel Abdalla, Olivier Chevassut, Pierre-Alain Fouque, and David Pointcheval. A simple threshold authenticated key exchange from short secrets. In Bimal K. Roy, editor, *ASIACRYPT 2005*, volume 3788 of *LNCS*, pages 566–584. Springer, Heidelberg, December 2005. (Cited on pages 57 and 58.)

[ACGP11] Michel Abdalla, Céline Chevalier, Louis Granboulan, and David Pointcheval. Contributory password-authenticated group key exchange with join capability. In Aggelos Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 142–160. Springer, Heidelberg, February 2011. (Not cited.)

[ACMP10] Michel Abdalla, Céline Chevalier, Mark Manulis, and David Pointcheval. Flexible group key exchange with on-demand computation of subgroup keys. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 351–368. Springer, Heidelberg, May 2010. (Not cited.)

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009. (Cited on pages 14, 15, 16, 17, 31, 42, 43, 49, 50, 51, 54, and 79.)

[AKB07] Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *NDSS 2007*. The Internet Society, February / March 2007. (Cited on page 50.)

[AP06] Michel Abdalla and David Pointcheval. A scalable password-based group key exchange protocol in the standard model. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 332–347. Springer, Heidelberg, December 2006. (Cited on page 66.)

[BBC$^+$13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, Heidelberg, February / March 2013. (Cited on pages 7, 16, 20, 33, 34, 41, 51, 54, 55, and 95.)

[BBC$^+$13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013. (Cited on pages 7, 16, 19, 20, 32, 34, 36, 49, 50, 54, and 137.)

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, Heidelberg, August 2004. (Cited on page 49.)

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of UC-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *ACNS 15*, volume 9092 of *LNCS*, pages 65–86. Springer, Heidelberg, June 2015. (Cited on pages 17, 18, 20, 44, 49, 62, 63, 64, 65, 66, 68, 72, 73, and 74.)

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, December 2016. (Cited on pages 7, 17, 19, 20, 34, 37, 40, 44, 49, 50, 65, 66, 74, 81, and 247.)

[BCFP10] Xavier Boyen, Céline Chevalier, Georg Fuchsbauer, and David Pointcheval. Strong cryptography from weak secrets. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 297–315. Springer, Heidelberg, May 2010. (Not cited.)

[BCG16] Olivier Blazy, Céline Chevalier, and Paul Germouty. Adaptive oblivious transfer and generalization. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 217–247. Springer, Heidelberg, December 2016. (Cited on pages 7, 17, 20, 61, 66, 68, 69, 72, and 207.)

[BCG17] Olivier Blazy, Céline Chevalier, and Paul Germouty. Almost optimal oblivious transfer from QA-NIZK. In Dieter Gollmann, Atsuko Miyaji, and Hiroaki Kikuchi, editors, *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, volume 10355 of *Lecture Notes in Computer Science*, pages 579–598. Springer, 2017. (Cited on pages 19, 20, 73, 74, and 75.)

[BCL$^+$05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 361–377. Springer, Heidelberg, August 2005. (Cited on pages 15, 26, 52, and 54.)

[BCNP04] Boaz Barak, Ran Canetti, Jesper Buus Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *45th FOCS*, pages 186–195. IEEE Computer Society Press, October 2004. (Cited on page 28.)

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 534–551. Springer, Heidelberg, June 2013. (Cited on pages 18, 20, 40, 41, and 42.)

[BCTV16] Fabrice Benhamouda, Céline Chevalier, Adrian Thillard, and Damien Vergnaud. Easing Coppersmith methods using analytic combinatorics: Applications to public-key cryptography with weak pseudorandomness. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 36–66. Springer, Heidelberg, March 2016. (Not cited.)

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-interactive zero-knowledge proofs of non-membership. In Kaisa Nyberg, editor, *CT-RSA 2015*, volume 9048 of *LNCS*, pages 145–164. Springer, Heidelberg, April 2015. (Cited on pages 18 and 35.)

[BCV16] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Mitigating server breaches in password-based authentication: Secure and efficient solutions. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 3–18. Springer, Heidelberg, February / March 2016. (Cited on pages 19, 20, 57, 58, 60, and 79.)

[BDS$^+$03] Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society, 2003. (Cited on page 50.)

[Bea96] Donald Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *28th ACM STOC*, pages 629–638. ACM Press, May 1996. (Cited on page 39.)

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001. (Cited on page 74.)

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012. (Cited on page 13.)

[BJKS03] John G. Brainard, Ari Juels, Burt Kaliski, and Michael Szydlo. A new two-server approach for authentication with short secrets. In *Proceedings of the 12th USENIX Security Symposium, Washington, D.C., USA, August 4-8, 2003*, 2003. (Cited on page 57.)

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (Hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, Heidelberg, August 2014. (Cited on pages 68 and 74.)

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Cited on pages 13, 15, and 57.)

[BN06] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *SAC 2005*, volume 3897 of *LNCS*, pages 319–331. Springer, Heidelberg, August 2006. (Cited on page 73.)

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, Heidelberg, May 2000. (Cited on pages 15, 19, 36, 58, 73, and 79.)

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, Heidelberg, March 2012. (Cited on pages 31, 69, and 70.)

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93*, pages 62–73. ACM Press, November 1993. (Cited on page 28.)

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer, Heidelberg, August 1994. (Cited on page 25.)

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001. (Cited on pages 14, 23, 24, 40, 57, 61, 67, and 71.)

[CC11] Hervé Chabanne and Céline Chevalier. *Vaudenay's Privacy Model in the Universal Composability Framework: A Case Study*, pages 16–24. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. (Not cited.)

[CCGS10] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 255–276. Springer, Heidelberg, August 2010. (Cited on pages 16 and 51.)

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 3–22. Springer, Heidelberg, August 2015. (Cited on pages 27, 61, and 68.)

[CDH12] Jan Camenisch, Maria Dubovitskaya, and Kristiyan Haralambiev. Efficient structure-preserving signature scheme from standard assumptions. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 76–94. Springer, Heidelberg, September 2012. (Cited on page 15.)

[CDK11] Céline Chevalier, Stéphanie Delaune, and Steve Kremer. Transforming password protocols to compose. In Supratik Chakraborty and Amit Kumar, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, volume 13 of *LIPIcs*, pages 204–216. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011. (Not cited.)

[CDKR13] Céline Chevalier, Stéphanie Delaune, Steve Kremer, and Mark Dermot Ryan. Composition of password-based protocols. *Formal Methods in System Design*, 43(3):369–413, 2013. (Not cited.)

[CDVW12] Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 449–466. Springer, Heidelberg, May 2012. (Cited on pages 19, 73, 74, and 80.)

[CEN15] Jan Camenisch, Robert R. Enderlein, and Gregory Neven. Two-server password-authenticated secret sharing UC-secure against transient corruptions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 283–307. Springer, Heidelberg, March / April 2015. (Cited on pages 57 and 58.)

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001. (Cited on pages 14, 15, 16, 39, 40, 42, and 49.)

[CFPZ09] Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal randomness extraction from a Diffie-Hellman element. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 572–589. Springer, Heidelberg, April 2009. (Not cited.)

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995. (Cited on page 13.)

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982. (Cited on page 23.)

[CHK$^+$05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, Heidelberg, May 2005. (Cited on pages 15, 26, 28, 47, 52, 74, and 79.)

[CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Heidelberg, May 2010. (Cited on page 65.)

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 453–474. Springer, Heidelberg, May 2001. (Cited on page 25.)

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, Heidelberg, February / March 2013. (Cited on pages 15, 17, 36, 61, 62, 65, 67, 71, and 74.)

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002. (Cited on pages 15, 16, 40, and 42.)

[CLV16] Céline Chevalier, Fabien Laguillaumie, and Damien Vergnaud. Privately outsourcing exponentiation to a single server: Cryptanalysis and optimal constructions. In Ioannis G. Askoxylakis, Sotiris Ioannidis, Sokratis K. Katsikas, and Catherine A. Meadows, editors, *ESORICS 2016, Part I*, volume 9878 of *LNCS*, pages 261–278. Springer, Heidelberg, September 2016. (Not cited.)

[CNs07] Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 573–590. Springer, Heidelberg, May 2007. (Cited on page 15.)

[CR03] Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 265–281. Springer, Heidelberg, August 2003. (Cited on page 26.)

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, Heidelberg, August 1998. (Cited on page 18.)

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002. (Cited on pages 14, 18, 31, 44, 63, 65, and 79.)

[CS03] Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, August 2003. (Cited on page 40.)

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Cited on page 39.)

[DG06] Mario Di Raimondo and Rosario Gennaro. Provably secure threshold password-authenticated key exchange. *J. Comput. Syst. Sci.*, 72(6):978–1001, 2006. (Cited on page 57.)

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on page 15.)

[DN02] Ivan Damgård and Jesper Buus Nielsen. Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 581–596. Springer, Heidelberg, August 2002. (Cited on pages 15 and 40.)

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. (Cited on pages 17, 18, 44, and 49.)

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984. (Cited on page 14.)

[FK00] Warwick Ford and Burton S. Kaliski Jr. Server-assisted generation of a strong secret from a password. In *9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2000), 4-16 June 2000, Gaithersburg, MD, USA*, pages 176–180, 2000. (Cited on page 57.)

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 468–485. Springer, Heidelberg, December 2011. (Cited on pages 15, 17, 18, 34, 36, 37, 40, 43, 44, 49, 65, and 79.)

[GD14] Vandana Guleria and Ratna Dutta. Lightweight universally composable adaptive oblivious transfer. In ManHo Au, Barbara Carminati, and C.-C.Jay Kuo, editors, *Network and System Security*, volume 8792 of *Lecture Notes in Computer Science*, pages 285–298. Springer International Publishing, 2014. (Cited on page 15.)

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer, Heidelberg, December 2007. (Cited on pages 15, 17, 66, and 68.)

[GH08] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer, Heidelberg, December 2008. (Cited on page 67.)

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, Heidelberg, May 2003. (Cited on pages 14, 15, 31, 33, 57, 66, and 79.)

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. (Cited on page 13.)

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, Heidelberg, April 2008. (Cited on pages 34, 40, and 49.)

[GWZ09] Juan A. Garay, Daniel Wichs, and Hong-Sheng Zhou. Somewhat non-committing encryption and efficient adaptively secure oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 505–523. Springer, Heidelberg, August 2009. (Cited on page 74.)

[Har11] Kristiyan Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications.* PhD thesis, New York University, 2011. (Cited on pages 16, 42, 63, and 65.)

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. (Cited on page 72.)

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 339–352. Springer, Heidelberg, August 1995. (Cited on page 58.)

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, Heidelberg, August 2007. (Cited on page 15.)

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, Heidelberg, February 2004. (Cited on pages 28, 40, and 65.)

[HO09] Brett Hemenway and Rafail Ostrovsky. Lossy trapdoor functions from smooth homomorphic hash proof systems. *Electronic Colloquium on Computational Complexity (ECCC)*, 16:127, 2009. (Cited on page 65.)

[Jab01] David P. Jablon. Password authentication using multiple servers. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 344–360. Springer, Heidelberg, April 2001. (Cited on page 57.)

[JL09a] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, Heidelberg, March 2009. (Cited on page 15.)

[JL09b] Stanislaw Jarecki and Xiaomin Liu. Private mutual authentication and conditional oblivious transfer. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 90–107. Springer, Heidelberg, August 2009. (Cited on page 50.)

[JR12] Charanjit S. Jutla and Arnab Roy. Relatively-sound NIZKs and password-based key-exchange. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 485–503. Springer, Heidelberg, May 2012. (Cited on page 49.)

[JR14] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to uc-pake and more. Cryptology ePrint Archive, Report 2014/805, 2014. (Cited on page 50.)

[JR15] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to UC-PAKE and more. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 630–655. Springer, Heidelberg, November / December 2015. (Cited on pages 19 and 73.)

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, Heidelberg, May 2005. (Cited on page 31.)

[Kat07] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 115–128. Springer, Heidelberg, May 2007. (Cited on page 40.)

[Kil88] Joe Kilian. Founding cryptography on oblivious transfer. In *20th ACM STOC*, pages 20–31. ACM Press, May 1988. (Cited on page 13.)

[KLL$^+$15] Aggelos Kiayias, Nikos Leonardos, Helger Lipmaa, Kateryna Pavlyk, and Qiang Tang. Optimal rate private information retrieval from homomorphic encryption. *PoPETs*, 2015(2):222–243, 2015. (Cited on page 17.)

[KM06] Neal Koblitz and Alfred Menezes. Another look at “provable security”. II. (invited talk). In Rana Barua and Tanja Lange, editors, *INDOCRYPT 2006*, volume 4329 of *LNCS*, pages 148–175. Springer, Heidelberg, December 2006. (Cited on page 25.)

[KM14] Franziskus Kiefer and Mark Manulis. Distributed smooth projective hashing and its application to two-server password authenticated key exchange. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *ACNS 14*, volume 8479 of *LNCS*, pages 199–216. Springer, Heidelberg, June 2014. (Cited on page 57.)

[KMTG12] Jonathan Katz, Philip D. MacKenzie, Gelareh Taban, and Virgil D. Gligor. Two-server password-only authenticated key exchange. *J. Comput. Syst. Sci.*, 78(2):651–669, 2012. (Cited on pages 19, 57, and 58.)

[KNP11] Kaoru Kurosawa, Ryo Nojima, and Le Trieu Phong. Generic fully simulatable adaptive oblivious transfer. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 274–291. Springer, Heidelberg, June 2011. (Cited on page 15.)

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, Heidelberg, May 2001. (Cited on pages 13, 14, 15, 16, and 57.)

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 275–295. Springer, Heidelberg, August 2015. (Cited on page 35.)

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS 2000*. The Internet Society, February 2000. (Cited on page 18.)

[Kra05] Hugo Krawczyk. HMQV: A high-performance secure Diffie-Hellman protocol. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 546–566. Springer, Heidelberg, August 2005. (Cited on page 25.)

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, Heidelberg, December 2009. (Cited on pages 15, 19, 65, and 73.)

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, March 2011. (Cited on pages 15, 16, 20, 36, 48, 49, 50, and 79.)

[KZ09] Aggelos Kiayias and Hong-Sheng Zhou. Zero-knowledge proofs with witness elimination. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, volume 5443 of *LNCS*, pages 124–138. Springer, Heidelberg, March 2009. (Cited on pages 18 and 19.)

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 182–189. ACM, July 2003. (Cited on pages 69 and 70.)

[Lin11a] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 446–466. Springer, Heidelberg, May 2011. (Cited on pages 15, 18, 20, 40, and 79.)

[Lin11b] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. Cryptology ePrint Archive, Report 2011/180, 2011. (Cited on page 41.)

[LY12] Benoît Libert and Moti Yung. Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 75–93. Springer, Heidelberg, March 2012. (Cited on page 49.)

[MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012. (Cited on page 65.)

[MSJ02] Philip D. MacKenzie, Thomas Shrimpton, and Markus Jakobsson. Threshold password-authenticated key exchange. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 385–400. Springer, Heidelberg, August 2002. (Cited on page 57.)

[Ngu05] Minh-Huyen Nguyen. The relationship between password-authenticated key exchange and other cryptographic primitives. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 457–475. Springer, Heidelberg, February 2005. (Cited on page 74.)

[NP97] Moni Naor and Benny Pinkas. Visual authentication and identification. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 322–336. Springer, Heidelberg, August 1997. (Cited on page 15.)

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001. (Cited on page 15.)

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990. (Cited on page 49.)

[OY91] Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In Luigi Logrippo, editor, *10th ACM PODC*, pages 51–59. ACM, August 1991. (Cited on page 58.)

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992. (Cited on pages 14, 16, and 42.)

[Poi12] David Pointcheval. Password-based authenticated key exchange (invited talk). In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 390–397. Springer, Heidelberg, May 2012. (Cited on page 15.)

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, Heidelberg, August 2008. (Cited on pages 15 and 74.)

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981. (Cited on pages 13 and 15.)

[RKP09] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 231–247. Springer, Heidelberg, August 2009. (Cited on page 15.)

[Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999. (Cited on page 49.)

[Sho99] Victor Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM, 1999. (Cited on page 25.)

[SK05] Michael Szydlo and Burton S. Kaliski Jr. Proofs for two-server password authentication. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 227–244. Springer, Heidelberg, February 2005. (Cited on page 57.)

[WHC$^+$14] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: Oblivious RAM for secure computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 191–202. ACM Press, November 2014. (Cited on page 13.)

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd FOCS*, pages 160–164. IEEE Computer Society Press, November 1982. (Cited on page 23.)

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986. (Cited on page 13.)

Part V

Appendices

Appendix A

Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages [BBC$^+$13a]

This is the Full Version of the Extended Abstract that appears in the Proceedings of the 16th International Conference on Practice and Theory in Public-Key Cryptography (PKC '13) (26 February – 1 March 2013, Nara, Japan), Kaoru Kurosawa Ed., Springer-Verlag, LNCS 7778, pages 272–291.

Authors

Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, Damien Vergnaud

Abstract

Authenticated Key Exchange (AKE) protocols enable two parties to establish a shared, cryptographically strong key over an insecure network using various authentication means, such as cryptographic keys, short (*i.e.*, low-entropy) secret keys or *credentials*. In this paper, we provide a general framework, that encompasses several previous AKE primitives such as *(Verifier-based) Password-Authenticated Key Exchange* or *Secret Handshakes*, we call *LAKE* for *Language-Authenticated Key Exchange*.

We first model this general primitive in the *Universal Composability* (UC) setting. Thereafter, we show that the Gennaro-Lindell approach can efficiently address this goal. But we need *smooth projective hash functions* on new languages, whose efficient implementations are of independent interest. We indeed provide such hash functions for languages defined by combinations of linear pairing product equations.

Combined with an efficient commitment scheme, that is derived from the highly-efficient UC-secure Lindell's commitment, we obtain a very practical realization of Secret Handshakes, but also *Credential-Authenticated Key Exchange protocols*. All the protocols are UC-secure, in the standard model with a common reference string, under the classical Decisional Linear assumption.

1 Introduction

The main goal of an *Authenticated Key Exchange* (AKE) protocol is to enable two parties to establish a shared cryptographically strong key over an insecure network under the complete control of an adversary. AKE is one of the most widely used and fundamental cryptographic primitives. In order for AKE to be possible, the parties must have authentication means, *e.g.* (public or secret) cryptographic keys, short (*i.e.*, low-entropy) secret keys or *credentials* that satisfy a (public or secret) policy.

Motivation. PAKE, for *Password-Authenticated Key Exchange*, was formalized by Bellovin and Merritt [BM92] and followed by many proposals based on different cryptographic assumptions (see [ACP09, CCGS10] and references therein). It allows users to generate a strong cryptographic key based on a shared "human-memorable" (*i.e.* low-entropy) password without requiring a public-key infrastructure. In this setting, an adversary controlling all communication in the network should not be able to mount an off-line dictionary attack.

The concept of *Secret Handshakes* has been introduced in 2003 by Balfanz, Durfee, Shankar, Smetters, Staddon and Wong [BDS$^+$03] (see also [JL09, AKB07]). It allows two members of the same group to identify each other secretly, in the sense that each party reveals his affiliation to the other only if they are members of the same group. At the end of the protocol, the parties can set up an ephemeral session key for securing further communication between them and an outsider is unable to determine if the handshake succeeded. In case of failure, the players do not learn any information about the other party's affiliation.

More recently, *Credential-Authenticated Key Exchange* (CAKE) was presented by Camenisch, Casati, Groß and Shoup [CCGS10]. In this primitive, a common key is established if and only if a specific relation is satisfied between credentials hold by the two players. This primitive includes variants of PAKE and Secret Handshakes, and namely Verifier-based PAKE, where the client owns a password pw and the server knows a one-way transformation v of the password only. It prevents massive password recovering in case of server corruption. The two players eventually agree on a common high entropy secret if and only if pw and v match together, and off-line dictionary attacks are prevented for third-party players.

Our Results. We propose a new primitive that encompasses most of the previous notions of authenticated key exchange. It is closely related to CAKE and we call it LAKE, for *Language-Authenticated Key-Exchange*, since parties establish a common key if and only if they hold credentials that belong to specific (and possibly independent) languages. The definition of the primitive is more practice-oriented than the definition of CAKE from [CCGS10] but the two notions are very similar. In particular, the new primitive enables privacy-preserving authentication and key exchange protocols by allowing two members of the same group to secretly and privately authenticate to each other without revealing this group beforehand.

In order to define the security of this primitive, we use the UC framework and an appropriate definition for languages that permits to dissociate the public part of the policy, the private common information the users want to check and the (possibly independent) secret values each user owns that assess the membership to the languages. We provide an ideal functionality for LAKE and give efficient realizations of the new primitive (for a large family of languages) secure under classical mild assumptions, in the standard model (with a common reference string – CRS), with static corruptions.

We significantly improve the efficiency of several CAKE protocols [CCGS10] for specific languages and we enlarge the set of languages for which we can construct practical schemes. Notably, we obtain a very practical realization of Secret Handshakes and a Verifier-based Password-Authenticated Key Exchange.

Our Techniques. A general framework to design PAKE in the CRS model was proposed by Gennaro and Lindell [GL03] in 2003. This approach was applied to the UC framework by Canetti, Halevi, Katz, Lindell, and MacKenzie [CHK$^+$05], and improved by Abdalla, Chevalier and Pointcheval [ACP09]. It makes use of the *smooth projective hash functions* (SPHF), introduced by Cramer and Shoup [CS02]. Such a hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projection* key one can only compute the function on a special subset of its domain. Our first contribution is the description of smooth projective hash functions for new interesting languages: Abdalla, Chevalier and Pointcheval [ACP09] explained how to make disjunctions and conjunctions of languages, we study here languages defined by linear pairing product equations on committed values.

In 2011, Lindell [Lin11] proposed a highly-efficient commitment scheme, with a non-interactive opening algorithm, in the UC framework. We will not use it in black-box, but instead we will patch it to make the initial Gennaro and Lindell's approach to work, without zero-knowledge proofs [CHK$^+$05], using the equivocability of the commitment.

Language Definition. In [ACP09], Abdalla *et al.* already formalized languages to be considered for SPHF. But, in the following, we will use a more simple formalism, which is nevertheless more general: we consider any efficiently computable binary relation $\mathcal{R} : \{0,1\}^* \times \mathcal{P} \times \mathcal{S} \to \{0,1\}$, where the additional parameters $\mathsf{pub} \in \{0,1\}^*$ and $\mathsf{priv} \in \mathcal{P}$ define a language $L_{\mathcal{R}}(\mathsf{pub}, \mathsf{priv}) \subseteq \mathcal{S}$ of the words W such that $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1$:

- pub are public parameters;
- priv are private parameters the two players have in mind, and they should think to the same values: they will be committed to, but never revealed;
- W is the word the sender claims to know in the language: it will be committed to, but never revealed.

Our LAKE primitive, specific to two relations $\mathcal{R}_a$ and $\mathcal{R}_b$, will allow two users, Alice and Bob, owning a word $W_a \in L_{\mathcal{R}_a}(\mathsf{pub}, \mathsf{priv}_a)$ and $W_b \in L_{\mathcal{R}_b}(\mathsf{pub}, \mathsf{priv}_b)$ respectively, to agree on a session key under some specific conditions: they first both agree on the public parameter pub, Bob will think about priv'_a for his expected value of priv_a, Alice will do the same with priv'_b for priv_b; eventually, if $\mathsf{priv}'_a = \mathsf{priv}_a$ and $\mathsf{priv}'_b = \mathsf{priv}_b$, and if they both know words in the languages, then the key agreement will succeed. In case of failure, no information should leak about the reason of failure, except the inputs did not satisfy the relations $\mathcal{R}_a$ or $\mathcal{R}_b$, or the languages were not consistent.

We stress that each LAKE protocol will be specific to a pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ describing the way Alice and Bob will authenticate to each other. This pair of relations $(\mathcal{R}_a, \mathcal{R}_b)$ specifies the sets $\mathcal{P}_a$, $\mathcal{P}_b$ and $\mathcal{S}_a$, $\mathcal{S}_b$ (to which the private parameters and the words should respectively belong). Therefore, the formats of priv_a, priv_b and W_a and W_b are known in advance, but not their values. When $\mathcal{R}_a$ and $\mathcal{R}_b$ are clearly defined from the context (e.g., PAKE), we omit them in the notations. For example, these relations can formalize:

- Password authentication: The language is defined by $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1 \Leftrightarrow W = \mathsf{priv}$, and thus $\mathsf{pub} = \emptyset$. The classical setting of PAKE requires the players A and B to use the same password W, and thus we should have $\mathsf{priv}_a = \mathsf{priv}'_b = \mathsf{priv}_b = \mathsf{priv}'_a = W_a = W_b$;
- Signature authentication: $\mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1 \Leftrightarrow \mathsf{Verif}(\mathsf{pub}_1, \mathsf{pub}_2, W) = 1$, where $\mathsf{pub} = (\mathsf{pub}_1 = \mathsf{vk}, \mathsf{pub}_2 = M)$ and $\mathsf{priv} = \emptyset$. The word W is thus a signature of M valid under vk, both specified in pub;
- Credential authentication: we can consider any mix for vk and M in pub or priv, and even in W, for which the relation $\mathcal{R}$ verifies the validity of the signature. When M and vk are in priv or W, we achieve *affiliation-hiding* property.

In the two last cases, the parameter pub can thus consist of a message on which the user is expected to know a signature valid under vk: either the user knows the signing key and can generate the signature on the fly to run the protocol, or the user has been given signatures on some messages (credentials). As a consequence, we just assume that, after having publicly agreed on a common pub, the two players have valid words in the appropriate languages. The way they have obtained these words does not matter.

Following our generic construction, private elements will be committed using encryption schemes, derived from Cramer-Shoup's scheme, and will thus have to be first encoded as n-tuples of elements in a group $\mathbb{G}$. In the case of PAKE, authentication will check that a player knows an appropriate password. The relation is a simple equality test, and accepts for one word only. A random commitment (and thus of a random group element) will succeed with negligible probability. For signature-based authentication, the verification key can be kept secret, but the signature should be unforgeable and thus a random word W should quite unlikely satisfy the relation. We will often make this assumption on useful relations $\mathcal{R}$: for any pub, $\{(\mathsf{priv}, W) \in \mathcal{P} \times \mathcal{S}, \mathcal{R}(\mathsf{pub}, \mathsf{priv}, W) = 1\}$ is sparse (negligible) in $\mathcal{P} \times \mathcal{S}$, and *a fortiori* in the set $\mathbb{G}^n$ in which elements are first embedded.

2 Definitions

In this section, we first briefly recall the notations and the security notions of the basic primitives we will use in the rest of the paper, and namely public key encryption and signature. More formal definitions, together with the classical computational assumptions (CDH, DDH, and DLin) are provided in the Appendix A.1: A public-key encryption scheme is defined by four algorithms: $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, $c \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m; r)$, and $m \leftarrow \mathsf{Decrypt}(\mathsf{dk}, c)$. We will need the classical notion of IND-CCA security. A signature scheme is defined by four algorithms: $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; s)$, and $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$. We will need the classical notion of EUF-CMA security. In both cases, the global parameters param will be ignored, included in the CRS. We will furthermore make use of collision-resistant hash function families.

2.1 Universal Composability

Our main goal will be to provide protocols with security in the universal composability framework. The interested reader is referred to [Can01, CHK+05] for details. More precisely, we will work in the UC framework with joint state proposed by Canetti and Rabin [CR03] (with the CRS as the joint state). Since players are not individually authenticated, but just afterward if the credentials are mutually consistent with the two players' languages, the adversary will be allowed to interact on behalf of any player from the beginning of the protocol, either with the credentials provided by the environment (static corruption) or without (impersonation attempt). As with the Split Functionality [BCL+05], according to whom sends the first flow for a player, either the player itself or the adversary, we know whether this is an honest player or a dishonest player (corrupted or impersonation attempt, but anyway controlled by the adversary). Then, our goal will be to prove that the best an adversary can do is to try to play against one of the other players, as an honest player would do, with a credential it guessed or obtained in any possible way. This is exactly the so-called one-line dictionary attack when one considers PAKE protocols. In the adaptive corruption setting, the adversary could get complete access to the private credentials and the internal memory of an honest player, and then get control of it, at any time. But we will restrict to the static corruption setting in this paper. It is enough to deal with most of the concrete requirements: related credentials, arbitrary compositions, and forward-secrecy. To achieve our goal, for a UC-secure LAKE, we will use some other primitives which are secure in the classical setting only.

2.2 Commitment

Commitments allow a user to commit to a value, without revealing it, but without the possibility to later change his mind. It is composed of three algorithms: $\mathsf{Setup}(1^k)$ generates the system parameters, according to a security parameter k; $\mathsf{Commit}(\ell, m; r)$ produces a commitment c on the input message $m \in \mathcal{M}$ using the random coins $r \overset{\$}{\leftarrow} \mathcal{R}$, under the label ℓ, and the opening information d; while $\mathsf{Decommit}(\ell, c, m, d)$ opens the commitment c with the message m and the opening information d that proves the correct opening under the label ℓ.

Such a commitment scheme should be both *hiding*, which says that the commit phase does not leak any information about m, and *binding*, which says that the decommit phase should not be able to open to two different messages. Additional features will be required in the following, such as non-malleability, extractability, and equivocability. We also included a label ℓ, which can be empty or an additional public information that has to be the same in both the commit and the decommit phases. A labeled commitment that is both non-malleable and extractable can be instantiated by an IND-CCA labeled encryption scheme (see the Appendix A.1). We will use the Linear Cramer-Shoup encryption scheme [Sha07, CKP07]. We will then patch it, using a technique inspired from [Lin11], to make it additionally equivocable (see Section 3). It will have an interactive commit phase, in two rounds: $\mathsf{Commit}(\ell, m; r)$ and a challenge ε from the receiver, which will define an implicit full commitment to be open latter.

2.3 Smooth Projective Hash Functions

Smooth projective hash function (SPHF) systems have been defined by Cramer and Shoup [CS02] in order to build a chosen-ciphertext secure encryption scheme. They have thereafter been extended [GL03, ACP09, BPV12] and applied to several other primitives. Such a system is defined on a language L, with five algorithms:

- $\mathsf{Setup}(1^k)$ generates the system parameters, according to a security parameter k;
- $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, L, W)$ derives the projection key hp, possibly depending on a word W;
- $\mathsf{Hash}(\mathsf{hk}, L, W)$ outputs the hash value from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, L, W, w)$ outputs the hash value from the projection key and the witness w that $W \in L$.

The correctness of the scheme assures that if W is in L with w as a witness, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, L, W) = \mathsf{ProjHash}(\mathsf{hp}, L, W, w)$. In our setting, these hash values will belong to a group $\mathbb{G}$. The security is defined through two different notions: the *smoothness* property guarantees that if $W \notin L$, the hash value is *statistically* indistinguishable from a random element, even knowing hp; the *pseudo-randomness* property guarantees that even for a word $W \in L$, but without the knowledge of a witness w, the hash value is *computationally* indistinguishable from a random element, even knowing hp.

3 Double Linear Cramer-Shoup Encryption (DLCS)

As explained earlier, any IND-CCA labeled encryption scheme can be used as a non-malleable and extractable labeled commitment scheme: one could use the Cramer-Shoup encryption scheme (see the Appendix A.4), but we will focus on the DLin-based primitives, and thus the Linear Cramer-Shoup scheme (see the Appendix A.3), we call LCS. Committed/encrypted elements will either directly be group elements, or bit-strings on which we apply a reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$. In order to add the equivocability, one can use a technique inspired from [Lin11]. See the Appendix B for more details, but we briefly present the commitment scheme we will use in the rest of this paper in conjunction with SPHF.

$$
\boxed{\begin{array}{lll}
\mathsf{Commit}(\ell, M; r, s, a, b, t):\ \text{for } (r, s, a, b, t) \overset{\$}{\leftarrow} \mathbb{Z}_p^5 & & \\
\quad (\mathcal{C}, \mathcal{C}') \leftarrow \mathsf{DLCSCom}(\ell, M, 1_{\mathbb{G}}; r, s, a, b) & & \\
\quad \chi = \mathfrak{H}_K(\mathcal{C}'), \mathcal{C}'' = g_1^t \zeta^\chi & \xrightarrow{\mathcal{C}, \mathcal{C}''} & \\
\quad \varepsilon \overset{?}{\neq} 0 \bmod p & \xleftarrow{\varepsilon} & \varepsilon \overset{\$}{\leftarrow} \mathbb{Z}_p^* \\
\quad \boldsymbol{z} = (z_r = r + \varepsilon a \bmod p, z_s = s + \varepsilon b \bmod p) & & \\
\mathsf{Decommit}(\ell, \mathcal{C}, \mathcal{C}', \varepsilon): & & \\
 & \xrightarrow{\mathcal{C}', t} & \chi = \mathfrak{H}_K(\mathcal{C}'), \mathcal{C}'' \overset{?}{=} g_1^t \zeta^\chi \\
\multicolumn{3}{r}{\text{With } \mathbf{z} = (z_r, z_s), \text{ implicit check of } \mathcal{C} \cdot \mathcal{C}'^{\varepsilon} \overset{?}{=} \mathsf{LCS}^*(\ell, \mathsf{ek}, M, \xi; z_r, z_s)}
\end{array}}
$$

Fig. 1. $\mathsf{DLCSCom}'$ Commitment Scheme for SPHF

Linear Cramer-Shoup Commitment Scheme. The parameters, in the CRS, are a group $\mathbb{G}$ of prime order p, with three independent generators $(g_1, g_2, g_3) \overset{\$}{\leftarrow} \mathbb{G}^3$, a collision-resistant hash function $\mathfrak{H}_K$, and possibly an additional reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$ to commit bit-strings. From 9 scalars $(x_1, x_2, x_3,\ y_1, y_2, y_3,\ z_1, z_2, z_3) \overset{\$}{\leftarrow} \mathbb{Z}_p^9$, one also sets, for $i = 1, 2$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. The public parameters consist of the encryption key $\mathsf{ek} = (\mathbb{G}, g_1, g_2, g_3, c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$, while the trapdoor for extraction is $\mathsf{dk} = (x_1, x_2, x_3,$ $y_1, y_2, y_3,\ z_1, z_2, z_3)$. One can define the encryption process:

$$\mathsf{LCS}(\ell, \mathsf{ek}, M; r, s) \overset{\text{def}}{=} (\mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^\xi)^r (c_2 d_2^\xi)^s)$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$. When ξ is specified from outside, one denotes it $\mathsf{LCS}^*(\ell, \mathsf{ek}, M, \xi; r, s)$. The commitment to a message $M \in \mathbb{G}$, or $M = \mathcal{G}(m)$ for $m \in \{0,1\}^n$, encrypts M under ek: $\mathsf{LCSCom}(\ell, M; r, s) \overset{\text{def}}{=} \mathsf{LCS}(\ell, \mathsf{ek}, M; r, s)$. The decommit process consists of M and (r, s) to check the correctness of the encryption. It is possible to do implicit verification, without any decommit information, but just an SPHF on the language of the ciphertexts of M that is privately shared by the two players. Since the underlying encryption scheme is IND-CCA, this commitment scheme is non-malleable and extractable.

Double Linear Cramer-Shoup Commitment Schemes. To make it equivocable, we double the commitment process, in two steps. The CRS additionally contains a scalar $\aleph \overset{\$}{\leftarrow} \mathbb{Z}_p$, one also sets, $\zeta = g_1^\aleph$. The trapdoor for equivocability is $\aleph$. The Double Linear Cramer-Shoup encryption scheme, denoted DLCS and detailed in the Appendix B is

$$\mathsf{DLCS}(\ell, \mathsf{ek}, M, N; r, s, a, b) \overset{\text{def}}{=} (\mathcal{C} \leftarrow \mathsf{LCS}(\ell, \mathsf{ek}, M; r, s), \mathcal{C}' \leftarrow \mathsf{LCS}^*(\ell, \mathsf{ek}, N, \xi; a, b))$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ is computed during the generation of $\mathcal{C}$ and transfered for the generation of $\mathcal{C}'$. As above, we denote $\mathsf{DLCSCom}$ denotes the use of DLCS with the encryption key ek. The usual commit/decommit processes are described on Figure 6 in the Appendix B. On Figure 1, one can find the $\mathsf{DLCSCom}'$ scheme where one can implicitly check the opening with an SPHF. These two constructions essentially differ with $\chi = \mathfrak{H}_K(\mathcal{C}')$ (for the SPHF implicit check) instead of $\chi = \mathfrak{H}_K(M, \mathcal{C}')$ (for the explicit check). We stress that with this alteration, the $\mathsf{DLCSCom}'$ scheme is not a real commitment scheme (not formally extractable/binding): in $\mathsf{DLCSCom}'$, the sender can indeed encrypt M in $\mathcal{C}$ and $N \neq 1_{\mathbb{G}}$ in $\mathcal{C}'$, and then, the global ciphertext $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$ contains $M' = MN^\varepsilon \neq M$, whereas one would have extracted M from $\mathcal{C}$. But M' is unknown before ε is sent, and thus, if one checks the membership of M' to a sparse language, it will unlikely be true.

Multi-Message Schemes. One can extend these encryption and commitment schemes to vectors of n messages (see the Appendix B). We will denote them n-$\mathsf{DLCSCom}'$ or n-$\mathsf{DLCSCom}$ for the commitment schemes. They consist in encrypting each message with independent random coins in $\mathcal{C}_i = (\mathbf{u}_i, e_i, v_i)$ but the same $\xi = \mathfrak{H}_K(\ell, (\mathbf{u}_i), (e_i))$, together with independent companion ciphertexts $\mathcal{C}'_i$ of $1_{\mathbb{G}}$, still with the same ξ for the doubled version. In the latter case, n independent challenges $\varepsilon_i \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ are then sent to lead to the full commitment $(\mathcal{C}_i \cdot \mathcal{C}_i'^{\varepsilon_i})$ with random coins $z_{r_i} = r_i+\varepsilon_i a_i$ and $z_{s_i} = s_i+\varepsilon_i b_i$. Again, if one of the companion ciphertext $\mathcal{C}'_i$ does not encrypt $1_{\mathbb{G}}$, the full commitment encrypts a vector with at least one unpredictable component M'_i. Several non-unity components in the companion ciphertexts would lead to independent components in the full commitment. For languages sparse enough, this definitely turns out not to be in the language.

4 SPHF for Implicit Proofs of Membership

In [ACP09], Abdalla *et al.* presented a way to compute a conjunction or a disjunction of languages by some simple operations on their projection keys. Therefore all languages presented afterward can easily be combined together. However as the original set of manageable languages was not really developed, we are going to present several steps to extend it, and namely in order to cover some languages useful in various AKE instantiations.

We will show that almost all the vast family of languages covered by the Groth-Sahai methodology [GS08] can be addressed by our approach too. More precisely, we can handle all the linear pairing product equations, when witnesses are committed using our above (multi-message) $\mathsf{DLCSCom}'$ commitment scheme, or even the non-equivocable LCSCom version. This will be strong enough for our applications. For using them in black-box to build our LAKE protocol, one should note that the projection key is computed from the ciphertext $\mathcal{C}$ when using the simple LCSCom commitment, but also when using the $\mathsf{DLCSCom}'$ version. The full commitment $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$ is not required, but ξ only, which is known as soon as $\mathcal{C}$ is given (or the vector $(\mathcal{C}_i)_i$ for the multi-message version). Of course, the hash value will then depend on the full commitment (either $\mathcal{C}$ for the LCSCom commitment, or $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$ for the $\mathsf{DLCSCom}'$ commitment).

This will be relevant to our AKE problem: equality of two passwords, in PAKE protocols; corresponding signing/verification keys associated with a valid signature on a pseudonym or a hidden identity, in secret handshakes; valid credentials, in CAKE protocols. All those tests are quite similar: one has to show that the ciphertexts are valid and that the plaintexts satisfy the expected relations in a group. We first illustrate that with commitments of Waters signatures of a public message under a committed verification key. We then explain the general method. The formal proofs are provided in the Appendix C.

4.1 Commitments of Signatures

Let us consider the Waters signature [Wat05] in a symmetric bilinear group, as reviewed in the Appendix A.3, and then we just need to recall that, in a pairing-friendly setting $(p, \mathbb{G}, \mathbb{G}_T, e)$, with public parameters $(\mathcal{F}, g, h)$, and a verification key vk, a signature $\sigma = (\sigma_1, \sigma_2)$ is valid with respect to the message M under the key vk if it satisfies $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

A similar approach has already been followed in [BPV12], however not with a Linear Cramer-Shoup commitment scheme, nor with such general languages. We indeed first consider the language of the signatures $(\sigma_1, \sigma_2) \in \mathbb{G}^2$ of a message $M \in \{0,1\}^k$ under the verification key $\mathsf{vk} \in \mathbb{G}$, where M is public but vk is private: $L(\mathsf{pub}, \mathsf{priv})$, where $\mathsf{priv} = \mathsf{vk}$ and $\mathsf{pub} = M$. One will thus commit the pair $(\mathsf{vk}, \sigma_1) \in \mathbb{G}^2$ with the label $\ell = (M, \sigma_2)$ using a 2-$\mathsf{DLCSCom}'$ commitment and then prove the commitment actually contains (vk, σ_1) such that $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$. We insist on the fact that σ_1 only has to be encrypted, and not σ_2, in order to hide the signature, since the latter σ_2 is a random group element. If one wants unlinkability between signature commitments, one simply needs to re-randomize (σ_1, σ_2) before encryption. Hence σ_2 can be sent in clear, but bounded to the commitment in the label, together with the pub part of the language. In order to prove the above property on the committed values, we will use conjunctions of SPHF:

first, to show that each commitment is well-formed (valid ciphertexts), and then that the associated plaintexts verify the linear pairing equation, where the committed values are underlined: $e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$ Note that vk is not used as a committed value for this verification of the membership of σ to the language since this is the verification key expected by the verifier, specified in the private part priv, which has to be independently checked with respect to the committed verification key. This is enough for the affiliation-hiding property. We could consider the similar language where $M \in \{0,1\}^k$ is in the word too: $e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\underline{\mathcal{F}(M)}, \sigma_2)$, and then one should commit M, bit-by-bit, and then use a $(k+2)$-$\mathsf{DLCSCom}'$ commitment.

4.2 Linear Pairing Product Equations

Instead of describing in details the SPHF for the above examples, let us show it for a more general framework: we considered

$$e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2) \text{ or } e(\underline{\sigma_1}, g) = e(h, \mathsf{vk}) \cdot e(\underline{\mathcal{F}(M)}, \sigma_2),$$

where the unknowns are underlined. These are particular instantiations of t simultaneous equations

$$\left(\prod_{i \in A_k} e(\underline{\mathcal{Y}_i}, \mathcal{A}_{k,i})\right) \cdot \left(\prod_{i \in B_k} \underline{\mathcal{Z}_i}^{\mathfrak{z}_{k,i}}\right) = \mathcal{B}_k, \text{ for } k = 1, \ldots, t,$$

where $\mathcal{A}_{k,i} \in \mathbb{G}$, $\mathcal{B}_k \in \mathbb{G}_T$, and $\mathfrak{z}_{k,i} \in \mathbb{Z}_p$, as well as $A_k \subseteq \{1, \ldots, m\}$ and $B_k \subseteq \{m+1, \ldots, n\}$ are public, but the $\mathcal{Y}_i \in \mathbb{G}$ and $\mathcal{Z}_i \in \mathbb{G}_T$ are simultaneously committed using the multi-message $\mathsf{DLCSCom}'$ or LCSCom commitments scheme, in $\mathbb{G}$ or $\mathbb{G}_T$ respectively. This is more general than the relations covered by [CCGS10], since one can also commit scalars bit-by-bit. In the Appendix C.4, we detail how to build the corresponding SPHF, and prove the soundness of our approach. For the sake of clarity, we focus here to a single equation only, since multiple equations are just conjunctions. We can even consider the simpler equation $\prod_{i=1}^{i=m} \underline{\mathcal{Z}_i}^{\mathfrak{z}_i} = \mathcal{B}$, since one can lift any ciphertext from $\mathbb{G}$ to a ciphertext in $\mathbb{G}_T$, setting $\mathcal{Z}_i = e(\mathcal{Y}_i, \mathcal{A}_i)$, as well as, for $j = 1,2,3$, $G_{i,j} = e(g_j, \mathcal{A}_i)$ and for $j = 1,2$, $H_{i,j} = e(h_j, \mathcal{A}_i)$, $C_{i,j} = e(c_j, \mathcal{A}_i)$, $D_{i,j} = e(d_j, \mathcal{A}_i)$, to lift all the group basis elements. Then, one transforms $\mathcal{C}_i = \mathsf{LCS}^*(\ell, \mathsf{ek}, \mathcal{Y}_i, \xi; \mathbf{z}_i) = (\boldsymbol{u_i} = (g_1^{z_{r_i}}, g_2^{z_{s_i}}, g_3^{z_{r_i}+z_{s_i}}), e_i = h_1^{z_{r_i}} h_2^{z_{s_i}} \cdot \mathcal{Y}_i, v_i = (c_1 d_1^{\xi})^{z_{r_i}} \cdot (c_2 d_2^{\xi})^{z_{s_i}})$ into $(\boldsymbol{U_i} = (G_{i,1}^{z_{r_i}}, G_{i,2}^{z_{s_i}}, G_{i,3}^{z_{r_i}+z_{s_i}}), E_i = H_{i,1}^{z_{r_i}} H_{i,2}^{z_{s_i}} \cdot \mathcal{Z}_i, V_i = (C_{i,1} D_{i,1}^{\xi})^{z_{r_i}} \cdot (C_{i,2} D_{i,2}^{\xi})^{z_{s_i}})$. Encryptions of $\mathcal{Z}_i$ originally in $\mathbb{G}_T$ use constant basis elements for $j = 1,2,3$, $G_{i,j} = G_j = e(g_j, g)$ and for $j = 1,2$, $H_{i,j} = H_j = e(h_j, g)$, $C_{i,j} = C_j = e(c_j, g)$, $D_{i,j} = D_j = e(d_j, g)$.

The commitments have been generated in $\mathbb{G}$ and $\mathbb{G}_T$ simultaneously using the m-$\mathsf{DLCSCom}'$ version, with a common ξ, where the possible combination with the companion ciphertext to the power ε leads to the above $\mathcal{C}_i$, thereafter lifted to $\mathbb{G}_T$. For the hashing keys, one picks random scalars $(\lambda, (\eta_i, \theta_i, \kappa_i, \mu_i)_{i=1,\ldots,m}) \xleftarrow{\$} \mathbb{Z}_p^{4m+1}$, and sets $\mathsf{hk}_i = (\eta_i, \theta_i, \kappa_i, \lambda, \mu_i)$. One then computes the projection keys as $\mathsf{hp}_i = (g_1^{\eta_i} g_3^{\kappa_i} h_1^{\lambda} (c_1 d_1^{\xi})^{\mu_i}, g_2^{\theta_i} g_3^{\kappa_i} h_2^{\lambda} (c_2 d_2^{\xi})^{\mu_i}) \in \mathbb{G}^2$. The hash value is

$$\prod_i e(u_{i,1}^{\eta_i} \cdot u_{i,2}^{\theta_i} \cdot u_{i,3}^{\kappa_i} \cdot e_i^{\lambda} \cdot v_i^{\mu_i}, \mathcal{A}_i) \cdot \mathcal{B}^{-\lambda} = \prod_i e(\mathsf{hp}_{i,1}^{z_{r_i}} \mathsf{hp}_{i,2}^{z_{s_i}}, \mathcal{A}_i),$$

where $\mathcal{A}_i$ is the constant used to compute $\mathcal{Z}_i = e(\mathcal{Y}_i, \mathcal{A}_i)$ and to lift ciphertexts from $\mathbb{G}$ to $\mathbb{G}_T$, or $\mathcal{A}_i = g^{\mathfrak{z}_i}$ if the ciphertext was already in $\mathbb{G}_T$. These evaluations can be computed either from the commitments and the hashing keys, or from the projection keys and the witnesses. We insist on the fact that, whereas the hash values are in $\mathbb{G}_T$, the projection keys are in $\mathbb{G}$ even if the ciphertexts are initially in $\mathbb{G}_T$. We stress again that the projection keys require the knowledge of ξ only: known from the LCSCom commitment or the first part $\mathcal{C}$ of the $\mathsf{DLCSCom}'$ commitment.

5 Language-Authenticated Key Exchange

5.1 The Ideal Functionality

We generalize the Password-Authenticated Key Exchange functionality $\mathcal{F}_{\text{PAKE}}$ (first provided in [CHK$^+$05]) to more complex languages: the players agree on a common secret key if and only if they own words that lie in the languages the partners have in mind. More precisely, after an agreement on pub between P_i and P_j (modeled here by the use of the split functionality, see below), player P_i uses a word W_i belonging to $L_i = L_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and it expects its partner P_j to use a word W_j belonging to the language $L'_j = L_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}'_j)$, and vice-versa for P_j and P_i. We assume relations $\mathcal{R}_i$ and $\mathcal{R}_j$ to be specified by the kind of protocol we study (PAKE, Verifier-based PAKE, secret handshakes, . . .) and so the languages are defined by the additional parameters pub, priv_i and priv_j only: they both agree on the public part pub, to be possibly parsed in a different way by each player for each language according to the relations. Note however that the respective languages do not need to be the same or to use similar relations: authentication means could be totally different for the 2 players. The key exchange should succeed if and only if the two following pairs of equations hold: ($L'_i = L_i$ and $W_i \in L_i$) and ($L'_j = L_j$ and $W_j \in L_j$).

Description. In the initial $\mathcal{F}_{\text{PAKE}}$ functionality [CHK$^+$05], the adversary was given access to a $\mathsf{TestPwd}$-query, which modeled the on-line dictionary attack. But it is known since [BCL$^+$05] that it is equivalent to use the split functionality model [BCL$^+$05], generate the $\mathsf{NewSession}$-queries corresponding to the corrupted players and tell the adversary (on behalf of the corrupted player) whether the protocol should succeed or not. Both methods enable the adversary to try a credential for a player (on-line dictionary attack). The second method (that we use here) implies allowing $\mathcal{S}$ to ask $\mathsf{NewSession}$-queries on behalf of the corrupted player, and letting it to be aware of the success or failure of the protocol in this case: the adversary learns this information only when it plays on behalf of a player (corruption or impersonation attempt). This is any way an information it would learn at the end of the protocol. We insist that third parties will not learn whether the protocol succeeded or not, as required for secret handshakes. To this aim, the NewKey-query informs in this case the adversary whether the credentials are consistent with the languages or not. In addition, the split functionality model guarantees from the beginning which player is honest and which one is controlled by the adversary. This finally allows us to get rid of the $\mathsf{TestPwd}$-query. The $\mathcal{F}_{\text{LAKE}}$ functionality is presented in Figure 2 and the corresponding split functionality $s\mathcal{F}_{\text{LAKE}}$ in Figure 3, where the languages are formally described and compared using the pub and priv parts.

The security goal is to show that the best attack for the adversary is a basic trial execution with a credential of its guess or choice: the proof will thus consist in emulating any real-life attack by either a trial execution by the adversary, playing as an honest player would do, but with a credential chosen by the adversary or obtained in any way; or a denial of service, where the adversary is clearly aware that its behavior will make the execution fail.

5.2 A Generic UC-Secure LAKE Construction

Intuition. Using smooth projective hash functions on commitments, one can generically define a LAKE protocol as done in [ACP09]. The basic idea is to make the player commit to their private information (for the expected languages and the owned words), and eventually the smooth projective hash functions will be used to make implicit validity checks of the global relation.

To this aim, we use the commitments and associated smooth projective hash functions as described in Sections 3 and 4. More precisely, all examples of SPHF in Section 4 can be used on extractable commitments divided into one or two parts (the non-equivocable LCSCom or the equivocable $\mathsf{DLCSCom}'$ commitments, see Figure 1). The relations on the committed values will

The functionality $\mathcal{F}_{\text{LAKE}}$ is parametrized by a security parameter k and a public parameter pub for the languages. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\dots,P_n$ via the following queries:

– New Session: Upon receiving a query $(\mathsf{NewSession} : \mathsf{sid}, P_i, P_j, W_i, L_i = L(\mathsf{pub}, \mathsf{priv}_i), L'_j = L(\mathsf{pub}, \mathsf{priv}'_j))$ from P_i,
  - If this is the first $\mathsf{NewSession}$-query with identifier sid, record the tuple $(P_i, P_j, W_i, L_i, L'_j, \mathsf{initiator})$. Send $(\mathsf{NewSession}; \mathsf{sid}, P_i, P_j, \mathsf{pub}, \mathsf{initiator})$ to $\mathcal{S}$ and P_j.
  - If this is the second $\mathsf{NewSession}$-query with identifier sid and there is a record $(P_j, P_i, W_j, L_j, L'_i, \mathsf{initiator})$, record the tuple $(P_j, P_i, W_j, L_j, L'_i, \mathsf{initiator}, W_i, L_i, L'_j, \mathsf{receiver})$. Send $(\mathsf{NewSession}; \mathsf{sid}, P_i, P_j, \mathsf{pub}, \mathsf{receiver})$ to $\mathcal{S}$ and P_j.

– Key Computation: Upon receiving a query $(\mathsf{NewKey} : \mathsf{sid})$ from $\mathcal{S}$, if there is a record of the form $(P_i, P_j, W_i, L_i, L'_j, \mathsf{initiator}, W_j, L_j, L'_i, \mathsf{receiver})$ and this is the first NewKey-query for session sid, then
  - If $(L'_i = L_i$ and $W_i \in L_i)$ and $(L'_j = L_j$ and $W_j \in L_j)$, then pick a random key sk of length k and store $(\mathsf{sid}, \mathsf{sk})$. In addition, if one player is corrupted, send $(\mathsf{sid}, \mathsf{success})$ to the adversary.
  - Else, store $(\mathsf{sid}, \perp)$, and send $(\mathsf{sid}, \mathsf{fail})$ to the adversary if one player is corrupted.

– Key Delivery: Upon receiving a query $(\mathsf{SendKey} : \mathsf{sid}, P_i, \mathsf{sk})$ from $\mathcal{S}$, then
  - if there is a record of the form $(\mathsf{sid}, \mathsf{sk}')$, then, if both players are uncorrupted, output $(\mathsf{sid}, \mathsf{sk}')$ to P_i. Otherwise, output $(\mathsf{sid}, \mathsf{sk})$ to P_i.
  - if there is a record of the form $(\mathsf{sid}, \perp)$, then pick a random key sk' of length k and output $(\mathsf{sid}, \mathsf{sk}')$ to P_i.

Fig. 2. Ideal Functionality $\mathcal{F}_{\text{LAKE}}$

Given the functionality $\mathcal{F}_{\text{LAKE}}$, the split functionality $s\mathcal{F}_{\text{LAKE}}$ proceeds as follows:

– Initialization:
  - Upon receiving $(\mathsf{Init}, \mathsf{sid}, \mathsf{pub}_i)$ from party P_i, send $(\mathsf{Init}, \mathsf{sid}, P_i, \mathsf{pub}_i)$ to the adversary.
  - Upon receiving a message $(\mathsf{Init}, \mathsf{sid}, P_i, H, \mathsf{pub}, \mathsf{sid}_H)$ from $\mathcal{S}$, where $H = \{P_i, P_j\}$ is a set of party identities, check that P_i has already sent $(\mathsf{Init}, \mathsf{sid}, \mathsf{pub}_i)$ and that for all recorded $(H', \mathsf{pub}', \mathsf{sid}_{H'})$, either $H = H'$, $\mathsf{pub} = \mathsf{pub}'$ and $\mathsf{sid}_H = \mathsf{sid}_{H'}$ or H and H' are disjoint and $\mathsf{sid}_H \neq \mathsf{sid}_{H'}$. If so, record the pair $(H, \mathsf{pub}, \mathsf{sid}_H)$, send $(\mathsf{Init}, \mathsf{sid}, \mathsf{sid}_H, \mathsf{pub})$ to P_i, and invoke a new functionality $(\mathcal{F}_{\text{LAKE}}, \mathsf{sid}_H, \mathsf{pub})$ denoted as $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$ and with set of honest parties H.

– Computation:
  - Upon receiving $(\mathsf{Input}, \mathsf{sid}, m)$ from party P_i, find the set H such that $P_i \in H$, the public value pub recorded, and forward m to $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$.
  - Upon receiving $(\mathsf{Input}, \mathsf{sid}, P_j, H, m)$ from $\mathcal{S}$, such that $P_j \notin H$, forward m to $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$ as if coming from P_j.
  - When $\mathcal{F}_{\text{LAKE}}^{(H,\mathsf{pub})}$ generates an output m for party $P_i \in H$, send m to P_i. If the output is for $P_j \notin H$ or for the adversary, send m to the adversary.

Fig. 3. Split Functionality $\boldsymbol{s\mathcal{F}}_{\text{LAKE}}$

not be explicitly checked, since the values will never be revealed, but will be implicitly checked using SPHF. It is interesting to note that in both cases (one-part or two-part commitment), the projection key will only depend on the first part of the commitment.

As it is often the case in the UC setting, we need the initiator to use stronger primitives than the receiver. They both have to use non-malleable and extractable commitments, but the initiator will use a commitment that is additionally equivocable, the $\mathsf{DLCSCom}'$ in two parts $((\mathcal{C}_i, \mathcal{C}'_i)$ and $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i)$, while the receiver will only need the basic LCSCom commitment in one part $(\mathsf{Com}_j = \mathcal{C}_j)$.

As already explained, SPHF will be used to implicitly check whether $(L'_i = L_i$ and $W_i \in L_i)$ and $(L'_j = L_j$ and $W_j \in L_j)$. But since in our instantiations private parameters priv and words W will have to be committed, the structure of these commitments will thus be publicly known in advance: commitments of $\mathcal{P}$-elements and $\mathcal{S}$-elements. Section 6 discusses on the languages captured by our definition, and illustrates with some AKE protocols. However, while these $\mathcal{P}$ and $\mathcal{S}$ sets are embedded in $\mathbb{G}^n$ from some n, it might be important to prove that the committed values are actually in $\mathcal{P}$ and $\mathcal{S}$ (e.g., one can have to prove it commits bits, whereas messages

Execution between P_i and P_j, with session identifier sid.

– Preliminary Round: each user generates a pair of signing/verification keys $(\mathsf{SK}, \mathsf{VK})$ and sends VK together with its contribution to the public part of the language.

We denote by $\ell_i = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{VK}_i, \mathsf{VK}_j)$ and by $\ell_j = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{VK}_j, \mathsf{VK}_i)$, where pub is the combination of the contributions of the two players. The initiator now uses a word W_i in the language $L(\mathsf{pub}, \mathsf{priv}_i)$, and the receiver uses a word W_j in the language $L(\mathsf{pub}, \mathsf{priv}_j)$, possibly re-randomized from their long-term secrets*. We assume commitments and associated smooth projective hash functions exist for these languages.

– First Round: user P_i (with random tape ω_i) generates a multi-$\mathsf{DLCSCom}'$ commitment on $(\mathsf{priv}_i, \mathsf{priv}'_j, W_i)$ in $(\mathcal{C}_i, \mathcal{C}'_i)$, where W_i has been randomized in the language, under the label ℓ_i. It also computes a Pedersen commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with random exponent t). It then sends $(\mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
– Second Round: user P_j (with random tape ω_j) computes a multi-LCS commitment on $(\mathsf{priv}_j, \mathsf{priv}'_i, W_j)$ in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\boldsymbol{r}$, where W_j has been randomized in the language, under the label ℓ_j. It then generates a challenge $\boldsymbol{\varepsilon}$ on $\mathcal{C}_i$ and hashing/projection keys** hk_i and hp_i associated to $\mathcal{C}_i$ (which will be associated to the future Com_i). It finally signs all the flows using SK_j in σ_j, and sends $(\mathcal{C}_j, \boldsymbol{\varepsilon}, \mathsf{hp}_i, \sigma_j)$ to P_i;
– Third Round: user P_i first checks the signature σ_j, computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\boldsymbol{\varepsilon}}_i$ and witness $\mathbf{z}$ (from $\boldsymbol{\varepsilon}$ and ω_i), it generates hashing/projection keys hk_j and hp_j associated to Com_j. It finally signs all the flows using SK_i in σ_i, and sends $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to P_j;
– Hashing: P_j first checks the signature σ_i and the correct opening of $\mathcal{C}''_i$ into $\mathcal{C}'_i$, it computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\boldsymbol{\varepsilon}}_i$.
  P_i computes K_i and P_j computes K_j as follows:

$$\begin{aligned} K_i &= \mathsf{Hash}(\mathsf{hk}_j, \{(\mathsf{priv}'_j, \mathsf{priv}_i)\} \times L(\mathsf{pub}, \mathsf{priv}'_j), \ell_j, \mathsf{Com}_j) \\ &\quad \times \mathsf{ProjHash}(\mathsf{hp}_i, \{(\mathsf{priv}_i, \mathsf{priv}'_j)\} \times L(\mathsf{pub}, \mathsf{priv}_i), \ell_i, \mathsf{Com}_i; \mathbf{z}) \\ K_j &= \mathsf{ProjHash}(\mathsf{hp}_j, \{(\mathsf{priv}_j, \mathsf{priv}'_i)\} \times L(\mathsf{pub}, \mathsf{priv}_j), \ell_j, \mathsf{Com}_j; \boldsymbol{r}) \\ &\quad \times \mathsf{Hash}(\mathsf{hk}_i, \{(\mathsf{priv}'_i, \mathsf{priv}_j)\} \times L(\mathsf{pub}, \mathsf{priv}'_i), \ell_i, \mathsf{Com}_i) \end{aligned}$$

* As explained in Section 1, recall that the languages considered depend on two possibly different relations, namely $L_i = L_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and $L_j = L_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}_j)$, but we omit them for the sake of clarity. We assume they are both self-randomizable.

** Recall that the SPHF is constructed in such a way that this projection key does not depend on $\mathcal{C}'_i$ and is indeed associated to the future whole Com_i.

Fig. 4. Language-based Authenticated Key Exchange from a Smooth Projective Hash Function on Commitments

are first embedded as group elements in $\mathbb{G}$ of large order p). This will be an additional language-membership to prove on the commitments.

This leads to a very simple protocol described on Figure 4. Note that if a player wants to make external adversaries think he owns an appropriate word, as it is required for Secret Handshakes, he can still play, but will compute everything with dummy words, and will replace the $\mathsf{ProjHash}$ evaluation by a random value, which will lead to a random key at the end.

Security Analysis. Since we have to assume common pub, we make a first round (with flows in each direction) where the players send their contribution, to come up with pub. These flows will also be used to know if there is a player controlled by the adversary (as with the Split Functionality [BCL$^+$05]). In case the languages have empty pub, these additional flows are not required, since the Split Functionality can be applied on the committed values. The signing key for the receiver is not required anymore since there is one flow only from its side. This LAKE protocol is secure against static corruptions. The proof is provided in the Appendix D, and is in the same vein as the one in [CHK$^+$05, ACP09]. However, it is a bit more intricate:

– in PAKE, when one is simulating a player, and knows the adversary used the correct password, one simply uses this password for the simulated player. In LAKE, when one knows the language expected by the adversary for the simulated player and has to simulate a successful execution (because of success announced by the NewKey-query), one has to actually include a correct word in the commitment: smooth projective hash functions do not allow the simulator to cheat, equivocability of the commitment is the unique trapdoor, but with a valid word. The languages must allow the simulator to produce a valid word W in $L(\mathsf{pub}, \mathsf{priv})$, for any pub and $\mathsf{priv} \in \mathcal{P}$ provided by the adversary or the environment. This will be the case in all the interesting applications of our protocol (see Section 6): if priv defines a Waters' verification key $\mathsf{vk} = g^x$, with the master key s such that $h = g^s$, the signing key is $\mathsf{sk} = h^x = \mathsf{vk}^s$, and thus the simulator can sign any message; if such a master key does not exist, one can restrict $\mathcal{P}$, and implicitly check it with the SPHF (the additional language-membership check, as said above). But since a random word is generated by the simulator, we need the real player to derive a random word from his own word, and the language to be *self-randomizable*.
– In addition, as already noted, our commitment $\mathsf{DLCSCom}'$ is not formally binding (contrarily to the much less efficient one used in [ACP09]). The adversary can indeed make the extraction give $\boldsymbol{M}$ from $\mathcal{C}_i$, whereas Com_i will eventually contain $\boldsymbol{M}'$ if $\mathcal{C}'_i$ does not encrypt $(1_{\mathbb{G}})^n$. However, since the actual value $\boldsymbol{M}'$ depends on the random challenge $\boldsymbol{\varepsilon}$, and the language is assumed sparse (otherwise authentication is easy), the protocol will fail: this can be seen as a denial of service from the adversary.

Theorem 1. *Our LAKE scheme from Figure 4 realizes the $s\mathcal{F}_{\text{LAKE}}$ functionality in the $\mathcal{F}_{\text{CRS}}$-hybrid model, in the presence of static adversaries, under the* DLin *assumption and the security of the One-Time Signature.*

Actually, from a closer look at the full proof, one can notice that $\mathsf{Com}_j = \mathcal{C}_j$ needs to be extractable, but $\mathsf{IND-CPA}$ security is enough, which leads to a shorter ciphertext (2 group elements less if one uses a Linear ciphertext instead of LCS). Similarly, one will not have to extract W_i from $\mathcal{C}_i$ when simulating sessions where P_i is corrupted. As a consequence, only the private parts of the languages have to be committed to in Com_i in the first and third rounds, whereas W_i can be encrypted independently with an $\mathsf{IND-CPA}$ encryption scheme in the third round only (5 group elements less in the first round, and 2 group elements less in the third round if one uses a Linear ciphertext instead of LCS).

6 Concrete Instantiations and Comparisons

In this section, we first give some concrete instantiations of several AKE protocols, using our generic protocol of LAKE, and compare the efficiencies of those instantiations.

6.1 Possible Languages

As explained above, our LAKE protocol is provably secure for *self-randomizable* languages only. While this notion may seem quite strong, most of the usual languages fall into it. For example, in a PAKE or a Verifier-based PAKE scheme, the languages consist of a single word and so trivially given a word, each user is able to deduce all the words in the language. One may be a little more worried about Waters Signature in our Secret Handshake, and/or Linear pairing equations. However the *self-randomizability* of the languages is easy to show:

– Given a Waters signature $\sigma = (\sigma_1, \sigma_2)$ over a message m valid under a verification key vk, one is able to randomize the signature into any signature over the same message m valid under the same verification key vk simply by picking a random s and computing $\sigma' = (\sigma_1 \cdot \mathcal{F}(m)^s, \sigma_2 \cdot g^s)$.

P_i uses a password W_i and P_j uses a password W_j. We denote $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$.

- First Round: P_i (with random tape ω_i) first generates a pair of signing/verification keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ and a $\mathsf{DLCSCom}'$ commitment on W_i in $(\mathcal{C}_i, \mathcal{C}'_i)$, under $\ell_i = (\ell, \mathsf{VK}_i)$. It also computes a Pedersen commitment on $\mathcal{C}'_i$ in $\mathcal{C}''_i$ (with random exponent t). It then sends $(\mathsf{VK}_i, \mathcal{C}_i, \mathcal{C}''_i)$ to P_j;
- Second Round: P_j (with random tape ω_j) computes a LCSCom commitment on W_j in $\mathsf{Com}_j = \mathcal{C}_j$, with witness $\boldsymbol{r}$, under the label ℓ. It then generates a challenge ε on $\mathcal{C}_i$ and hashing/projection keys hk_i and the corresponding hp_i for the equality test on Com_i ("Com_i is a valid commitment of W_j", this only requires the value ξ_i computable thanks to $\mathcal{C}_i$). It then sends $(\mathcal{C}_j, \varepsilon, \mathsf{hp}_i)$ to P_i;
- Third Round: user P_i can compute $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$ and witness $\mathbf{z}$ (from ε and ω_i), it generates hashing/projection keys hk_j and hp_j for the equality test on Com_j. It finally signs all the flows using SK_i in σ_i and sends $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to P_j;
- Hashing: P_j first checks the signature and the validity of the Pedersen commitment (thanks to t), it computes $\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$. P_i computes K_i and P_j computes K_j as follows:

$$K_i = \mathsf{Hash}(\mathsf{hk}_j, L'_j, \ell, \mathsf{Com}_j) \cdot \mathsf{ProjHash}(\mathsf{hp}_i, L_i, \ell_i, \mathsf{Com}_i; \mathbf{z})$$
$$K_j = \mathsf{ProjHash}(\mathsf{hp}_j, L_j, \ell, \mathsf{Com}_j; \boldsymbol{r}) \cdot \mathsf{Hash}(\mathsf{hk}_i, L'_i, \ell_i, \mathsf{Com}_i)$$

Fig. 5. Password-based Authenticated Key Exchange

- For linear pairing equations, with public parameters $\mathcal{A}_i$ for $i = 1, \dots, m$ and γ_i for $i = m+1, \dots, n$, and $\mathcal{B}$, given $(\mathcal{X}_1, \dots, \mathcal{X}_m, \mathcal{Z}_{m+1}, \dots, \mathcal{Z}_n)$ verifying $\prod_{i=1}^m e(\mathcal{X}_i, \mathcal{A}_i) \cdot \prod_{i=m+1}^n \mathcal{Z}_i^{\gamma_i} = \mathcal{B}$, one can randomize the word in the following way:
  - If $m < n$, one simply picks random $(\mathcal{X}'_1, \dots, \mathcal{X}'_m)$, $(\mathcal{Z}'_{m+1}, \dots, \mathcal{Z}'_{n-1})$ and sets $\mathcal{Z}'_n = (\mathcal{B}/(\prod_{i=1}^m e(\mathcal{X}'_i, \mathcal{A}_i) \cdot \prod_{i=m+1}^{n-1} \mathcal{Z}'^{\gamma_i}_i))^{1/\gamma_n}$,
  - Else, if $m = n > 1$, one picks random $r_1, \dots, r_{n-1}$ and sets $\mathcal{X}'_i = \mathcal{X}_i \cdot \mathcal{A}_n^{r_i}$, for $i = 1, \dots, m-1$ and $\mathcal{X}'_m = \mathcal{X}_m \cdot \prod_{i=1}^{m-1} \mathcal{A}_i^{-r_i}$,
  - Else $m = n = 1$, this means only one word satisfies the equation. So we already have this word.

As we can see most of the common languages manageable with a SPHF are already *self-randomizable*. We now show how to use them in concrete instantiations.

6.2 Concrete Instantiations

Password-Authenticated Key Exchange. Using our generic construction, we can easily obtain a PAKE protocol, as described on Figure 5, where we optimize from the generic construction, since $\mathsf{pub} = \emptyset$, removing the agreement on pub, but still keeping the one-time signature keys $(\mathsf{SK}_i, \mathsf{VK}_i)$ to avoid man-in-the-middle attacks since it has another later flow: P_i uses a password W_i and expects P_j to own the same word, and thus in the language $L'_j = L_i = \{W_i\}$; P_j uses a password W_j and expects P_i to own the same word, and thus in the language $L'_i = L_j = \{W_j\}$; The relation is the equality test between priv_i and priv_j, which both have no restriction in $\mathbb{G}$ (hence $\mathcal{P} = \mathbb{G}$). As the word W_i, the language private parameters priv_i of a user and priv'_j of the expected language for the other user are the same, each user can commit in the protocol to only one value: its password.

We kept the general description and notations in Figure 5, but $\mathcal{C}_j$ can be a simply $\mathsf{IND-CPA}$ encryption scheme. It is quite efficient and relies on the DLin assumption, with DLCS for $(\mathcal{C}_i, \mathcal{C}'_i)$ and thus 10 group elements, but a Linear encryption for $\mathcal{C}_j$ and thus 3 group elements. Projection keys are both 2 group elements. Globally, P_i sends 13 groups elements plus 1 scalar, a verification key and a one-time signature, while P_j sends 5 group elements and 1 scalar: 18 group elements and 2 scalars in total. We can of course instantiate it with the Cramer-Shoup and ElGamal variants, under the DDH assumption: P_i sends 8 groups elements plus 1 scalar, a verification key and a one-time signature, while P_j sends 3 group elements and 1 scalar (all group elements can be in the smallest group): 11 group elements and 2 scalars in total.

Verifier-based PAKE. The above scheme can be modified into an efficient PAKE protocol that is additionally secure against *server compromise*: the so-called verifier-based PAKE, where the client owns a password pw, while the server knows a verifier only, such as g^{pw}, so that in case of break-in to the server, the adversary will not immediately get all the passwords.

To this aim, as usually done, one first does a PAKE with g^{pw} as common password, then asks the client to additionally prove it can compute the Diffie-Hellman value h^{pw} for a basis h chosen by the server. Ideally, we could implement this trick, where the client P_j just considers the equality test between the g^{pw} and the value committed by the server for the language $L'_i = L_j$, while the server P_i considers the equality test with $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, where h is sent as its contribution to the public part of the language by the server $L_i = L'_j$. Since the server chooses h itself, it chooses it as $h = g^{\alpha}$, for an ephemeral random α, and can thus compute $h^{\mathsf{pw}} = (g^{\mathsf{pw}})^{\alpha}$. On its side, the client can compute this value since it knows pw. The client could thus commit to $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, in order to prove its knowledge of pw, whereas the server could just commit to g^{pw}. Unfortunately, from the extractability of the server commitment, one would just get g^{pw}, which is not enough to simulate the client.

To make it in a provable way, the server chooses an ephemeral h as above, and they both run the previous PAKE protocol with $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$ as common password, and mutually checked: h is seen as the pub part, hence the preliminary flows are required.

Credential-Authenticated Key Exchange. In [CCGS10], the authors proposed instantiations of the CAKE primitive for conjunctions of atomic policies that are defined algebraically by relations of the form $\prod_{j=1}^{k} g_j^{F_j} = 1$ where the g_j's are elements of an abelian group and F_j's are integer polynomials in the variables committed by the users.

The core of their constructions relies on their practical UC zero-knowledge proof. There is no precise instantiation of such proof, but it is very likely to be inefficient. Their proof technique indeed requires to transform the underlying Σ-protocols into corresponding Ω-protocols [GMY06] by verifiably encrypting the witness. An Ω-protocol is a Σ-protocol with the additional property that it admits a polynomial-time straight-line extractor. Since the witnesses are scalars in their algebraic relations, their approach requires either inefficient bit-per-bit encryption of these witnesses or Paillier encryption in which case the problem of using group with different orders in the representation and in the encryption requires additional overhead.

Even when used with Σ-protocols, their PAKE scheme without UC-security, requires at least two proofs of knowledge of representations that involve at least 30 group elements (if we assume the encryption to be linear Cramer Shoup), and some extra for the last proof of existence (*cf.* [CKS11]), where our PAKE requires less than 20 group elements. Anyway they say, their PAKE scheme is less efficient than [CHK+05], which needed 6 rounds and around 30 modular exponentiations per user, while our efficient PAKE requires less than 40 exponentiations, in total, in only 3 rounds. Our scheme is therefore more efficient than the scheme from [CHK+05] for the same security level (*i.e.* UC-security with static corruptions).

Secret-Handshakes. We can also instantiate a (linkable) Secret Handshakes protocol, using our scheme with two different languages: P_i will commit to a valid signature σ_i on a message m_i (his identity for example), under a private verification key vk_i, and expects P_j to commit to a valid signature on a message m'_j under a private verification key vk'_j; but P_j will do analogously with a signature σ_j on m_j under vk_j, while expecting a signature on m'_i under vk'_i. The public parts of the signature (the second component) are sent in clear with the commitments.

In a regular Secret Handshakes both users should use the same languages. But here, we have a more general situation (called *dynamic matching* in [AKB07]): the two participants will have the same final value if and only if they both belong to the organization the other expects. If one lies, our protocol guarantees no information leakage. Furthermore, the semantic security of the session is even guaranteed with respect to the authorities, in a forward-secure way (this property is also achieved in [JL09] but in a weaker security model). Finally, our scheme supports revocation and can handle roles as in [AKB07].

Standard secret handshakes, like [AKB07], usually work with credentials delivered by a unique authority, this would remove our need for a hidden verification key, and private part of the language. Both users would only need to commit to signatures on their identity/credential, and show that they are valid. This would require a dozen of group elements with our approach. Their construction requires only 4 elements under BDH, however it relies on the asymmetric Waters IBE with only two elements, whereas the only security proof known for such IBE [Duc10] requires an extra term in $\mathbb{G}_2$ which would render their technique far less efficient, as several extra terms would be needed to expect a provably secure scheme. While sometimes less effective, our LAKE approach can manage Secret Handshakes, and provide additional functionalities, like more granular control on the credential as part of them can be expressly hidden by both the users. More precisely, we provide affiliation-hiding property and let third parties unaware of the success/failure of the protocol.

Unlinkable Secret-Handshakes. Moving the users' identity from the public pub part to individual private priv part, and combining our technique with [BPV12], it is also possible to design an *unlinkable* Secret Handshakes protocol [JL09] with practical efficiency. It illustrates the case where committed values have to be proven in a strict subset of $\mathbb{G}$, as one has to commit to bits: the signed message M is now committed and not in clear, it thus has to be done bit-by-bit since the encoding $\mathcal{G}$ does not allow algebraic operations with the content to apply the Waters function on the message. It is thus possible to prove the knowledge of a Waters signature on a private message (identity) valid under a private verification key. Additional relations can be required on the latter to make authentication even stronger.

Acknowledgments

This work was supported in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet and the ICT Program under Contract ICT-2007-216676 ECRYPT II.

References

ACGP11. Michel Abdalla, Céline Chevalier, Louis Granboulan, and David Pointcheval. Contributory password-authenticated group key exchange with join capability. In Aggelos Kiayias, editor, *Topics in Cryptology – CT-RSA 2011*, volume 6558 of *Lecture Notes in Computer Science*, pages 142–160. Springer, February 2011.

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, August 2009.

AKB07. Giuseppe Ateniese, Jonathan Kirsch, and Marina Blanton. Secret handshakes with dynamic and fuzzy matching. In *ISOC Network and Distributed System Security Symposium – NDSS 2007*. The Internet Society, February / March 2007.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.

BCL$^+$05. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377. Springer, August 2005.

BDS$^+$03. Dirk Balfanz, Glenn Durfee, Narendar Shankar, Diana K. Smetters, Jessica Staddon, and Hao-Chi Wong. Secret handshakes from pairing-based key agreements. In *IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Computer Society, 2003.

BFPV11. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 403–422. Springer, March 2011.

BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111, Springer, March 2012.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

CCGS10. Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 255–276. Springer, August 2010.

CHK+05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005.

CKP07. Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 613–630. Springer, August 2007.

CKS11. Jan Camenisch, Stephan Krenn, and Victor Shoup. A framework for practical universally composable zero-knowledge protocols. In *Advances in Cryptology – ASIACRYPT 2011*, Lecture Notes in Computer Science, pages 449–467. Springer, December 2011.

CR03. Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 265–281. Springer, August 2003.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, April / May 2002.

Duc10. Léo Ducas. Anonymity from asymmetry: New constructions for anonymous HIBE. In Josef Pieprzyk, editor, *Topics in Cryptology – CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 148–164. Springer, March 2010.

GL03. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

GMR88. Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

GMY06. Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. *Journal of Cryptology*, 19(2):169–209, April 2006.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

JL09. Stanislaw Jarecki and Xiaomin Liu. Private mutual authentication and conditional oblivious transfer. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 90–107. Springer, August 2009.

Lin11. Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 446–466. Springer, May 2011.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, August 1992.

Sha07. Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/2007/074.pdf`.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.

A Preliminaries

A.1 Formal Definitions of the Primitives

We first recall the definitions of the basic tools, with the security notions with success/advantage that all depend on a security parameter (which is omitted here for simplicity of notation).

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions $\mathfrak{H}_K$ from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^k$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any adversary $\mathcal{A}$ on a random function $\mathfrak{H}_K \overset{\$}{\leftarrow} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\begin{aligned}\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}) &= \Pr[\mathfrak{H}_K \overset{\$}{\leftarrow} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(\mathfrak{H}_K) : \mathfrak{H}_K(m_0) = \mathfrak{H}_K(m_1)],\\ \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) &= \max_{\mathcal{A} \leq t}\{\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A})\}.\end{aligned}$$

Labeled encryption scheme. A labeled public-key encryption scheme is defined by four algorithms:

– $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
– $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the encryption key ek and the decryption key dk;
– $\mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key ek, using the random coins r;
– $\mathsf{Decrypt}(\ell, \mathsf{dk}, c)$ outputs the plaintext m encrypted in c under the label ℓ, or $\perp$.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

– *Correctness*: for all key pair $(\mathsf{ek}, \mathsf{dk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)) = m.$$

– *Indistinguishability under chosen-ciphertext attacks*: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls FIND and GUESS, and makes use of the oracle $\mathsf{ODecrypt}$:
  • $\mathsf{ODecrypt}(\ell, c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key dk. The input queries (ℓ, c) are added to the list $\mathcal{CT}$.

```
$\mathsf{Exp}^{\mathsf{ind-cca}-b}_{\mathcal{E},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\ell^*, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek}, \mathsf{ODecrypt}(\cdot,\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\ell, \mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot,\cdot))$
6. IF $(\ell^*, c^*) \in \mathcal{CT}$ RETURN 0
7. ELSE RETURN $b'$
```

The advantages are

$$\begin{aligned}\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathcal{E}}(\mathcal{A}) &= \Pr[\mathsf{Exp}^{\mathsf{ind-cca}-1}_{\mathcal{E},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind-cca}-0}_{\mathcal{E},\mathcal{A}}(k) = 1]\\ \mathsf{Adv}^{\mathsf{ind-cca}}_{\mathcal{E}}(t) &= \max_{\mathcal{A} \leq t}\{\mathsf{Adv}^{\mathsf{ind-cca}}_{\mathcal{E}}(\mathcal{A})\}.\end{aligned}$$

Labeled commitment scheme. A labeled commitment scheme is defined by three algorithms:

– $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
– $\mathsf{Commit}(\ell, m; r)$ produces a commitment c and the opening information d on the input message $m \in \mathcal{M}$ under the label ℓ, using the random coins r;
– $\mathsf{Decommit}(\ell, c, m, d)$ checks the validity of the opening information d on the commitment c for the message m under the label ℓ. It answers 1 for true, and 0 for false.

A commitment scheme $\mathcal{C}$ should satisfy the following properties

– *Correctness*: for any label ℓ, and all messages m, if $(c, d) \leftarrow \mathsf{Commit}(\ell, m; r)$, then we have $\mathsf{Decommit}(\ell, c, m, d) = 1$.
– *Hiding*: this security notion is similar to the indistinguishability under chosen-plaintext attacks for encryption, which means that c does not help to distinguish between two candidates m_0 and m_1 as committed values.
– *Binding*: this security notion is more an unforgeability notion, which means that for any commitment c, it should be hard to open it in two different ways, which means to exhibit (m_0, d_0) and (m_1, d_1), such that $m_0 \neq m_1$ and $\mathsf{Decommit}(\ell, c, m_0, d_0) = \mathsf{Decommit}(\ell, c, m_1, d_1) = 1$.

The commitment algorithm can be interactive between the sender and the received, but the hiding and the binding properties should still hold. Several additional properties are sometimes required:

– *Extractability*: an indistinguishable Setup procedure also outputs a trapdoor that allows a extractor to get the committed value m from any commitment c. More precisely, if c can be open in a valid way, the extractor can get this value from the commitment.
– *Equivocability*: an indistinguishable Setup procedure also outputs a trapdoor that allows a simulator to generate commitments that can thereafter be open in any way.
– *Non-Malleability*: it should be hard, from a commitment c to generate a new commitment $c' \neq c$ whose committed values are in relation.

It is well-known that any IND-CCA encryption scheme leads to a non-malleable and extractable commitment scheme [GL03].

Signature scheme. A signature scheme is defined by four algorithms:

– $\mathsf{Setup}(1^k)$, where k is the security parameter, generates the global parameters param of the scheme;
– $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the verification key vk and the signing key sk;
– $\mathsf{Sign}(\mathsf{sk}, m; s)$ produces a signature σ on the input message m, under the signing key sk, and using the random coins s;
– $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.

A signature scheme $\mathcal{S}$ should satisfy the following properties

– *Correctness*: for all key pair $(\mathsf{vk}, \mathsf{sk})$, all random coins s and all messages m, we have the equality $\mathsf{Verif}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m; s)) = 1$.
– *Existential unforgeability under (adaptive) chosen-message attacks*: this security notion can be formalized by the following security game, where it makes use of the oracle OSign:

    • $\mathsf{OSign}(m)$: This oracle outputs a valid signature on m under the signing key sk. The input queries m are added to the list $\mathcal{SM}$.

```
$\mathsf{Exp}^{\mathsf{euf-cma}}_{\mathcal{S},\mathcal{A}}(k)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\cdot))$
4. $b \leftarrow \mathsf{Verif}(\mathsf{vk}, m^*, \sigma^*)$
5. IF $M \in \mathcal{SM}$ RETURN 0
6. ELSE RETURN $b$
```

The success probabilities are

$$\mathsf{Succ}_{\mathcal{S}}^{\mathsf{euf-cma}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathcal{S},\mathcal{A}}^{\mathsf{euf}}(k) = 1] \quad \mathsf{Succ}_{\mathcal{S}}^{\mathsf{euf-cma}}(k,t) = \max_{\mathcal{A} \leq t}\{\mathsf{Succ}_{\mathcal{S}}^{\mathsf{euf-cma}}(\mathcal{A})\}.$$

Smooth Projective Hash Function. A smooth projective hash function system is defined on a language L, with five algorithms:

- $\mathsf{Setup}(1^k)$ generates the system parameters, according to a security parameter k;
- $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, L, W)$ derives the projection key hp, possibly depending on a word W;
- $\mathsf{Hash}(\mathsf{hk}, L, W)$ outputs the hash value from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, L, W, w)$ outputs the hash value from the projection key and the witness w that $W \in L$.

The correctness of the scheme assures that if W is in L with w as a witness, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, L, W) = \mathsf{ProjHash}(\mathsf{hp}, L, W, w)$. In our setting, these hash values will belong to a group $\mathbb{G}$. The security is defined through two different notions, the smoothness and the pseudo-randomness properties, where we use the distribution $\Delta(L, W) = \{(\mathsf{hk}, \mathsf{hp}), \mathsf{hk} \leftarrow \mathsf{HashKG}(L), \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L, W)\}$:

- the *smoothness* property guarantees that if $W \notin L$, the hash value is *statistically* indistinguishable from a random element, even knowing hp:
$$\begin{aligned}&\{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \leftarrow \mathsf{Hash}(\mathsf{hk}, L, W)\}\\ &\approx_s \{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \overset{\$}{\leftarrow} \mathbb{G}\}.\end{aligned}$$
We define by $\mathsf{Adv}^{\mathsf{smooth}}$ the statistical distance between the two distributions.
- the *pseudo-randomness* property guarantees that even for a word $W \in L$, but without the knowledge of a witness w, the hash value is *computationally* indistinguishable from a random element, even knowing hp:
$$\begin{aligned}&\{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \leftarrow \mathsf{Hash}(\mathsf{hk}, L, W)\}\\ &\approx_c \{(\mathsf{hp}, G), (\mathsf{hk}, \mathsf{hp}) \leftarrow \Delta(L, W), G \overset{\$}{\leftarrow} \mathbb{G}\}.\end{aligned}$$
We define by $\mathsf{Adv}^{\mathsf{pr}}(t)$ the computational distance between the two distributions for t-time distinguishers.

A.2 Computational Assumptions

The three classical assumptions we use along this paper are: the computational Diffie-Hellman (CDH), the decisional Diffie-Hellman (DDH) and the decisional Linear (DLin) assumptions. Our constructions essentially rely on the DLin assumption, that implies the CDH. It is the most general since it (presumably) holds in many groups, with or without pairing. Some more efficient instantiations will rely on the DDH assumption but in more specific groups.

Definition 2 (Computational Diffie-Hellman (CDH)). *The Computational Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b) for unknown random $a, b \overset{\$}{\leftarrow} \mathbb{Z}_p$, it is hard to compute g^{ab}. We define by $\mathsf{Succ}_{p,\mathbb{G},g}^{\mathsf{cdh}}(t)$ the best advantage an adversary can have in finding g^{ab} within time t.*

Definition 3 (Decisional Diffie-Hellman (DDH)). *The Decisional Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b, g^c) for unknown random $a, b \overset{\$}{\leftarrow} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \overset{\$}{\leftarrow} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{ddh}}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple within time t.*

Definition 4 (Decisional Linear Problem (DLin)). *The Decisional Linear Problem [BBS04] says that, in a group $(p, \mathbb{G}, g)$, when we are given $(g^x, g^y, g^{xa}, g^{yb}, g^c)$ for unknown random $x, y, a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = a + b \bmod p$ (a linear tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in distinguishing a linear tuple from a random tuple within time t.*

A.3 Some Primitives in Symmetric Groups – Based on DLin

Linear Cramer-Shoup (LCS) encryption scheme. The Linear Cramer-Shoup encryption scheme [Sha07] can be tuned to a labeled public-key encryption scheme:

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with three independent generators $(g_1, g_2, g_3) \xleftarrow{\$} \mathbb{G}^3$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \xleftarrow{\$} \mathbb{Z}_p^9$, and sets, for $i = 1, 2$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r, s)$, for a message $M \in \mathbb{G}$ and two random scalars $r, s \xleftarrow{\$} \mathbb{Z}_p$, the ciphertext is $\mathcal{C} = (\mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^\xi)^r (c_2 d_2^\xi)^s)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathcal{C} = (\mathbf{u}, e, v))$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \cdot u_3^{x_3+\xi y_3} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^{z_1} u_2^{z_2} u_3^{z_3})$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DLin assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

Waters signature. The Waters signature [Wat05] is defined as follows:

- $\mathsf{Setup}(1^k)$: In a pairing-friendly setting $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, one chooses a random vector $\boldsymbol{f} = (f_0, \ldots, f_k) \xleftarrow{\$} \mathbb{G}^{k+1}$ that defines the Waters hash function $\mathcal{F}(M) = f_0 \prod_{i=1}^k f_i^{M_i}$ for $M \in \{0,1\}^k$, and an extra generator $h \xleftarrow{\$} \mathbb{G}$. The global parameters param consist of all these elements $(p, \mathbb{G}, g, \mathbb{G}_T, e, \boldsymbol{f}, h)$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public verification key as $\mathsf{vk} = g^x$, and the secret signing key as $\mathsf{sk} = h^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = \big(\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g^s\big)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g) \stackrel{?}{=} e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

This scheme is existentially unforgeable against (adaptive) chosen-message attacks [GMR88] under the CDH assumption.

A.4 Some Primitives in Asymmetric Groups – Based on DDH

Cramer-Shoup encryption scheme. The Cramer-Shoup encryption scheme [CS98] can be tuned into a labeled public-key encryption scheme:

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a collision-resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.

- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^z)$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DDH assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

Waters signature (asymmetric). This variant of the Waters signature has been proposed and proved in [BFPV11]:

- $\mathsf{Setup}(1^k)$: In a bilinear group $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, \mathfrak{g}_1, \mathbb{G}_T, e)$, one chooses a random vector $\boldsymbol{f} = (f_0, \dots, f_k) \stackrel{\$}{\leftarrow} \mathbb{G}_1^{k+1}$, an extra generator $h_1 \stackrel{\$}{\leftarrow} \mathbb{G}_1$. The global parameters param consist of $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, \mathfrak{g}_1, \mathbb{G}_T, e, \boldsymbol{f}, h_1)$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, which defines the public $\mathsf{vk} = \mathfrak{g}_1^x$, and the secret $\mathsf{sk} = h_1^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, $\sigma = \big(\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \boldsymbol{\sigma_2} = (g_1^s, \mathfrak{g}_1^s)\big)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, \mathfrak{g}_1) = e(h_1, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_{2,2})$, and $e(\sigma_{2,1}, \mathfrak{g}_1) = e(g_1, \sigma_{2,2})$.

This scheme is unforgeable under the following variant of the CDH assumption:

Definition 5 (The Advanced Computational Diffie-Hellman problem (CDH^+)). *In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, \mathfrak{g}_1, \mathbb{G}_T, e)$. The CDH^+ assumption states that given $(g_1, \mathfrak{g}_1, g_1^a, \mathfrak{g}_1^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.*

B Multi Double Linear Cramer-Shoup Commitment

B.1 Multi Double Linear Cramer-Shoup ($n - \mathsf{DLCS}$) Encryption

We can extend the encryption scheme implicitly presented in Section 3 to vectors $(M_i, N_i)_{i=1,\dots,n}$, partially IND-CCA protected, with a common ξ. It of course also includes the $n - \mathsf{LCS}$ scheme on vectors $(M_i)_i$, when ignoring the $\mathcal{C}'$ part, which is already anyway the case for the decryption oracle:

- $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with three independent generators $(g_1, g_2, g_3) \stackrel{\$}{\leftarrow} \mathbb{G}^3$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^9$, and sets, for $i = 1, 2$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. It also chooses a collision-resistant hash function $\mathfrak{H}_K$. The encryption key is $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, \boldsymbol{M}; \boldsymbol{r}, \boldsymbol{s})$, for a vector $\boldsymbol{M} \in \mathbb{G}^n$ and two vectors $\boldsymbol{r}, \boldsymbol{s} \in \mathbb{Z}_p^n$, computes

  $$\mathcal{C} = (\mathcal{C}_1, \dots, \mathcal{C}_n), \text{ where } \mathcal{C}_i = (\mathbf{u}_i = (g_1^{r_i}, g_2^{s_i}, g_3^{r_i+s_i}), e_i = M_i \cdot h_1^{r_i} h_2^{s_i}, v_i = (c_1 d_1^\xi)^{r_i} (c_2 d_2^\xi)^{s_i})$$

  with the v_i computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \dots, \mathbf{u}_n, e_1, \dots, e_n)$.
- $\mathsf{Encrypt}'(\ell, \mathsf{ek}, \boldsymbol{N}, \xi; \boldsymbol{a}, \boldsymbol{b})$, for a vector $\boldsymbol{N} \in \mathbb{G}^n$ and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^n$, computes

  $$\mathcal{C}' = (\mathcal{C}'_1, \dots, \mathcal{C}'_n), \text{ where } \mathcal{C}'_i = (\boldsymbol{\alpha}_i = (g_1^{a_i}, g_2^{b_i}, g_3^{a_i+b_i}), \beta_i = N_i \cdot h_1^{a_i} h_2^{b_i}, \gamma_i = (c_1 d_1^\xi)^{a_i} (c_2 d_2^\xi)^{b_i})$$

  where the γ_i's are computed with the above $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \dots, \mathbf{u}_n, e_1, \dots, e_n)$, hence the additional input.
  One can use both simultaneously: on input $(\ell, \mathsf{ek}, \boldsymbol{M}, \boldsymbol{N}; \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{b})$, the global encryption algorithm first calls $\mathsf{Encrypt}(\ell, \mathsf{ek}, \boldsymbol{M}; \boldsymbol{r}, \boldsymbol{s})$ and to get $\mathcal{C}$ and ξ, and then calls $\mathsf{Encrypt}'(\ell, \mathsf{ek}, \boldsymbol{N}, \xi; \boldsymbol{a}, \boldsymbol{b})$ to get $\mathcal{C}'$.

– $\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathcal{C}, \mathcal{C}')$: one first parses $\mathcal{C} = (C_1, \ldots, C_n)$ and $\mathcal{C}' = (\mathcal{C}'_1, \ldots, \mathcal{C}'_n)$, where $C_i = (\mathbf{u}_i, e_i, v_i)$ and $\mathcal{C}'_i = (\boldsymbol{\alpha}_i, \beta_i, \gamma_i)$, for $i = 1, \ldots, n$, computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$ and checks whether, for $i = 1, \ldots, n$, $u_{i,1}^{x_1+\xi y_1} \cdot u_{i,2}^{x_2+\xi y_2} \cdot u_{i,3}^{x_3+\xi y_3} \stackrel{?}{=} v_i$ (but not for the γ_i's). If the equality holds, one computes $M_i = e_i/(u_{i,1}^{z_1} u_{i,2}^{z_2} u_{i,3}^{z_3})$ and $N_i = \beta_i/(\alpha_{i,1}^{z_1} \alpha_{i,2}^{z_2} \alpha_{i,3}^{z_3})$, and outputs $(\boldsymbol{M} = (M_1, \ldots, M_n), \boldsymbol{N} = (N_1, \ldots, N_n))$. Otherwise, one outputs $\perp$.
– $\mathsf{PDecrypt}(\ell, \mathsf{dk}, \mathcal{C})$: is a partial decryption algorithm that does as above but working on the $\mathcal{C}$ part only to get $\boldsymbol{M} = (M_1, \ldots, M_n)$ or $\perp$.

DLCS denotes the particular case where $n = 1$: $\mathsf{DLCS}(\ell, \mathsf{ek}, M, N; r, s, a, b) = (\mathcal{C}, \mathcal{C}')$, with

$$\mathcal{C} = (\mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^\xi)^r (c_2 d_2^\xi)^s) = \mathsf{LCS}(\ell, \mathsf{ek}, M; r, s),$$
$$\mathcal{C}' = (\boldsymbol{\alpha} = (g_1^a, g_2^b, g_3^{a+b}), \beta = N \cdot h_1^a h_2^b, \gamma = (c_1 d_1^\xi)^a (c_2 d_2^\xi)^b) = \mathsf{LCS}^*(\ell, \mathsf{ek}, N, \xi; a, b)$$

where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.

B.2 Security of the Multi Double Linear Cramer Shoup Encryption

Security model. This scheme is indistinguishable against *partial-decryption chosen-ciphertext* attacks, where a partial-decryption oracle only is available, but even when we allow the adversary to choose $\boldsymbol{M}$ and $\boldsymbol{N}$ in two different steps (see the security game below), under the DLin assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

Indistinguishability against partial-decryption chosen-ciphertext attacks for vectors, in two steps: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls $\mathtt{FIND}_M$, $\mathtt{FIND}_N$ and $\mathtt{GUESS}$. In the first stage $\mathtt{FIND}_M$, it receives the encryption key ek; in the second stage $\mathtt{FIND}_N$, it receives the encryption of $\boldsymbol{M}_b$: $\mathcal{C}^* = \mathsf{Encrypt}(\ell, \mathsf{ek}, \boldsymbol{M}_b)$; in the last stage $\mathtt{GUESS}$ it receives the encryption of $\boldsymbol{N}_b$: $\mathcal{C}'^* = \mathsf{Encrypt}'(\ell, \mathsf{ek}, \xi^*, \boldsymbol{N}_b)$, where ξ^* is the value involved in $\mathcal{C}$. During all these stages, it can make use of the oracle $\mathsf{ODecrypt}(\ell, \mathcal{C})$, that outputs the decryption of $\mathcal{C}$ under the label ℓ and the challenge decryption key dk, using $\mathsf{PDecrypt}(\ell, \mathsf{dk}, \mathcal{C})$. The input queries $(\ell, \mathcal{C})$ are added to the list $\mathcal{CT}$.

```
$\mathsf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind-pd-cca}-b}(k, n)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$; $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
2. $(\ell^*, \boldsymbol{M}_0, \boldsymbol{M}_1) \leftarrow \mathcal{A}(\mathtt{FIND}_M : \mathsf{ek}, \mathsf{ODecrypt}(\cdot, \cdot))$
3. $\mathcal{C}^* \leftarrow \mathsf{Encrypt}(\ell^*, \mathsf{ek}, \boldsymbol{M}_b)$
4. $(\boldsymbol{N}_0, \boldsymbol{N}_1) \leftarrow \mathcal{A}(\mathtt{FIND}_N : \mathcal{C}^*, \mathsf{ODecrypt}(\cdot, \cdot))$
5. $\mathcal{C}'^* \leftarrow \mathsf{Encrypt}'(\ell^*, \mathsf{ek}, \xi^*, \boldsymbol{N}_b)$
6. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathcal{C}'^*, \mathsf{ODecrypt}(\cdot, \cdot))$
7. IF $(\ell^*, \mathcal{C}^*) \in \mathcal{CT}$ RETURN $0$
8. ELSE RETURN $b'$
```

The advantages are, where q_d is the number of decryption queries:

$$\mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind-pd-cca}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind-pd-cca}-1}(k, n) = 1] - \Pr[\mathsf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind-pd-cca}-0}(k, n) = 1]$$
$$\mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind-pd-cca}}(n, q_d, t) = \max_{\mathcal{A} \leq t} \mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind-pd-cca}}(\mathcal{A}).$$

Theorem 6. *The Multiple $n - \mathsf{DLCS}$ encryption scheme is IND-PD-CCA if $\mathcal{H}$ is a collision-resistant hash function family, under the DLin assumption in $\mathbb{G}$:*

$$\mathsf{Adv}_{n-\mathsf{DLCS}}^{\mathsf{ind-pd-cca}}(n, q_d, t) \leq 4n \times \left(\mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{dlin}}(t) + \mathsf{Succ}_{\mathcal{H}}^{\mathsf{coll}}(t) + \frac{q_d}{p} \right).$$

Corollary 7. *The Multiple $n - \mathsf{LCS}$ encryption scheme is IND-CCA if $\mathcal{H}$ is a collision-resistant hash function family, under the DLin assumption in $\mathbb{G}$.*

Security proof. Let us be given a DLin challenge $(g_1, g_2, g_3, u_1 = g_1^r, u_2 = g_2^s, u_3 = g_3^t)$, for which we have to decide whether (u_1, u_2, u_3) is a linear tuple in basis (g_1, g_2, g_3), and thus $t = r + s \bmod p$, or a random one. From an IND-PD-CCA adversary $\mathcal{A}$ against the encryption scheme, we built a DLin distinguisher $\mathcal{B}$. The latter first uses (g_1, g_2, g_3) as the global parameters. It also picks $x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3 \overset{\$}{\leftarrow} \mathbb{Z}_p^9$ and sets $c_i = g_i^{x_i} g_3^{x_3}, d_i = g_i^{y_i} g_3^{y_3}, h_i = g_i^{z_i} g_3^{z_3}$, for $i = 1, 2$. It chooses a collision-resistant hash function $\mathfrak{H}_K$ and provides $\mathcal{A}$ with the encryption key $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.

– In the initial game $\mathcal{G}_0$,
  - $\mathcal{A}$'s decryption queries are answered by $\mathcal{B}$, simply using the decryption key dk.
  - When $\mathcal{A}$ submits the first challenge vectors $\boldsymbol{M}_b = (M_{b,1}, \ldots, M_{b,n})$ for $b = 0, 1$, with a label ℓ^*, $\mathcal{B}$ chooses a random bit $b \overset{\$}{\leftarrow} \{0, 1\}$ and encrypts $\boldsymbol{M}_b$:
    * it chooses two random vectors $\boldsymbol{r}^*, \boldsymbol{s}^* \overset{\$}{\leftarrow} \mathbb{Z}_p^n$
    * it defines $\mathcal{C}_i^* = (\mathbf{u}_i^* = (g_1^{r_i^*}, g_2^{s_i^*}, g_3^{r_i^*+s_i^*}), e_i^* = M_{b,i} \cdot h_1^{r_i^*} h_2^{s_i^*}, v_i^* = (c_1 d_1^{\xi^*})^{r_i^*} (c_2 d_2^{\xi^*})^{s_i^*})$, for $i = 1, \ldots, n$, where the v_i^*'s are computed with the value $\xi^* = \mathfrak{H}_K(\ell^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_n^*, e_1^*, \ldots, e_n^*)$, and $\mathcal{C}^* = (\mathcal{C}_1^*, \ldots, \mathcal{C}_n^*)$.
  - When $\mathcal{A}$ submits the second challenge vectors $\boldsymbol{N}_b = (N_{b,1}, \ldots, N_{b,n})$ for $b = 0, 1$,
    * $\mathcal{B}$ chooses two random vectors $\boldsymbol{a}^*, \boldsymbol{b}^* \overset{\$}{\leftarrow} \mathbb{Z}_p^n$
    * it defines $\mathcal{C}_i'^* = (\boldsymbol{\alpha}_i^* = (g_1^{a_i^*}, g_2^{b_i^*}, g_3^{a_i^*+b_i^*}), \beta_i^* = N_{b,i} \cdot h_1^{a_i^*} h_2^{b_i^*}, \gamma_i^* = (c_1 d_1^{\xi^*})^{a_i^*} (c_2 d_2^{\xi^*})^{b_i^*})$, for $i = 1, \ldots, n$, where the γ_i^*'s are computed with the above $\xi^* = \mathfrak{H}_K(\ell^*, \mathbf{u}_1^*, \ldots, \mathbf{u}_n^*, e_1^*, \ldots, e_n^*)$, and $\mathcal{C}'^* = (\mathcal{C}_1'^*, \ldots, \mathcal{C}_n'^*)$.
  - When $\mathcal{A}$ returns b', $\mathcal{B}$ outputs $b' \overset{?}{=} b$.

$$\Pr_0[1 \leftarrow \mathcal{B}] = \Pr_0[b' = b] = (\mathsf{Adv}_{n-\mathsf{DLCS}}^{\mathsf{ind-pd-cca}}(\mathcal{A}) - 1)/2.$$

– In game $\mathcal{G}_1$, where we assume $t = r + s \bmod p$, to encrypt the challenge vectors $\boldsymbol{M}_b$ and $\boldsymbol{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}_1^*$: $\mathcal{C}_1^* = (\mathbf{u}_1^* = (u_1, u_2, u_3), e_1^* = M_{b,1} \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}, v_1^* = u_1^{x_1+\xi^* y_1} u_2^{x_2+\xi^* y_2} u_3^{x_3+\xi^* y_3})$, which actually defines $r_1^* = r$ and $s_1^* = s$.

$$\mathbf{u}_1^* = (g_1^{r_1^*}, g_2^{s_1^*}, g_3^{r_1^*+s_1^*}) \qquad e_1^* = M_{b,1} \cdot (g_1^{r_1^*})^{z_1} (g_2^{s_1^*})^{z_2} (g_3^{r_1^*+s_1^*})^{z_3} = M_{b,1} \cdot h_1^{r_1^*} h_2^{s_1^*}$$
$$v_1^* = (g_1^{r_1^*})^{x_1+\xi^* y_1} (g_2^{s_1^*})^{x_2+\xi^* y_2} (g_3^{r_1^*+s_1^*})^{x_3+\xi^* y_3} = (c_1 d_1^{\xi^*})^{r_1^*} (c_2 d_2^{\xi^*})^{s_1^*}$$

The challenge ciphertexts are identical to the encryptions of $\boldsymbol{M}_b$ and $\boldsymbol{N}_b$ in $\mathcal{G}_0$. Decryption queries are still answered the same way. Hence the gap between this game and the previous game is 0.

$$\Pr_1[1 \leftarrow \mathcal{B}] = \Pr_0[1 \leftarrow \mathcal{B}] = (\mathsf{Adv}_{n-\mathsf{DLCS}}^{\mathsf{ind-pd-cca}}(\mathcal{A}) - 1)/2.$$

– In game $\mathcal{G}_2$, we now assume that $t \overset{\$}{\leftarrow} \mathbb{Z}_p$ (a random tuple). First, we have to check that the *incorrect* computation of v_1^* does not impact the probability to reject invalid ciphertexts, then we prove that e_1^* is totally independent of $M_{b,1}$.

  1. About the validity checks, $u_{i,1}^{x_1+\xi y_1} \cdot u_{i,2}^{x_2+\xi y_2} \cdot u_{i,3}^{x_3+\xi y_3} \overset{?}{=} v_i$, where $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$, three cases can appear with respect to the challenge ciphertext equal to $\mathcal{C}^* = ((\mathbf{u}_1^*, e_1^*, v_1^*), \ldots, (\mathbf{u}_n^*, e_n^*, v_n^*))$:
     (a) $(\ell, \mathbf{u}_1, e_1, \ldots, \mathbf{u}_n, e_n) = (\ell^*, \mathbf{u}_1^*, e_1^*, \ldots, \mathbf{u}_n^*, e_n^*)$, then necessarily, for some i, $v_i \neq v_i^*$, then the check on index i will fail since one value only is acceptable;
     (b) $(\ell, \mathbf{u}_1, e_1, \ldots, \mathbf{u}_n, e_n) \neq (\ell^*, \mathbf{u}_1^*, e_1^*, \ldots, \mathbf{u}_n^*, e_n^*)$, but $\xi = \xi^*$, then the adversary has generated a collision for the hash function $\mathfrak{H}_K$.
     (c) $(\ell, \mathbf{u}_1, e_1, \ldots, \mathbf{u}_n, e_n) \neq (\ell^*, \mathbf{u}_1^*, e_1^*, \ldots, \mathbf{u}_n^*, e_n^*)$, and $\xi \neq \xi^*$: the ciphertext should be accepted iff $v_i = u_{i,1}^{x_1+\xi y_1} \cdot u_{i,2}^{x_2+\xi y_2} \cdot u_{i,3}^{x_3+\xi y_3}$, for $i = 1, \ldots, n$. To make it acceptable,

if we denote $g_2 = g_1^{\beta_2}$ and $g_3 = g_1^{\beta_3}$, we indeed have

$$\begin{array}{lllllll}
\log_{g_1} c_1 = & x_1 & & +\beta_3 x_3 & & & \\
\log_{g_1} d_1 = & & & & y_1 & & +\beta_3 y_3 \\
\log_{g_1} c_2 = & & \beta_2 x_2 & +\beta_3 x_3 & & & \\
\log_{g_1} d_2 = & & & & & \beta_3 y_2 & +\beta_3 y_3
\end{array}$$

with in addition,

$$\begin{aligned}
\log_{g_1} v_1^* &= r x_1 + s\beta_2 x_2 + t\beta_3 x_3 + r\xi^* y_1 + s\xi^* \beta_2 y_2 + t\xi^* \beta_3 y_3 \\
\log_{g_1} v_i^* &= r_i^* x_1 + s_i^* \beta_2 x_2 + (r_i^* + s_i^*)\beta_3 x_3 + r_i^* \xi^* y_1 + s_i^* \xi^* \beta_2 y_2 + (r_i^* + s_i^*)\xi^* \beta_3 y_3 \\
&= r_i^* \log_{g_1} c_1 + s_i^* \log_{g_1} c_2 + \xi^* r_i^* \log_{g_1} d_1 + \xi^* s_i^* \log_{g_1} c_2 \qquad \text{for } i = 2, \ldots, n \\
\log_{g_1} \gamma_i^* &= a_i^* x_1 + b_i^* \beta_2 x_2 + (a_i^* + b_i^*)\beta_3 x_3 + a_i^* \xi^* y_1 + b_i^* \xi^* \beta_2 y_2 + (a_i^* + b_i^*)\xi^* \beta_3 y_3 \\
&= a_i^* \log_{g_1} c_1 + b_i^* \log_{g_1} c_2 + \xi^* a_i^* \log_{g_1} d_1 + \xi^* b_i^* \log_{g_1} c_2 \qquad \text{for } i = 1, \ldots, n
\end{aligned}$$

The $2n-1$ last relations are thus linearly dependent with the 4 above relations, hence remains the useful relations

$$\begin{array}{llllllll}
\log_{g_1} c_1 = & x_1 & & +\beta_3 x_3 & & & & (1) \\
\log_{g_1} d_1 = & & & & y_1 & & +\beta_3 y_3 & (2) \\
\log_{g_1} c_2 = & & \beta_2 x_2 & +\beta_3 x_3 & & & & (3) \\
\log_{g_1} d_2 = & & & & & \beta_2 y_2 & +\beta_3 y_3 & (4) \\
\log_{g_1} v_1^* = & r x_1 & +s\beta_2 x_2 & +t\beta_3 x_3 & +r\xi^* y_1 & +s\xi^* \beta_2 y_2 & +t\xi^* \beta_3 y_3 & (5)
\end{array}$$

One can note that for v_1^* to be predictable, because of the x_1, x_2 and y_1, y_2 components, we need to have (5) $= r$ (1) $+ s$ (3) $+ r\xi^*$ (2) $+ s\xi^*$ (4), and then $t = r + s$, which is not the case, hence v_1^* looks random: in this game, v_1^* is perfectly uniformly distributed in $\mathbb{G}$.
Furthermore, for any v_i in the decryption query, if $\mathbf{u}_i = (g_1^{r'}, g_2^{s'}, g_3^{t'})$ is not a linear triple, then it should be such that

$$\log_{g_1} v_i = r' x_1 + s'\beta_2 x_2 + t'\beta_3 x_3 + r'\xi y_1 + s'\xi\beta_2 y_2 + t'\xi\beta_3 y_3.$$

Since the matrix

$$\begin{pmatrix}
1 & 0 & \beta_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & \beta_3 \\
0 & \beta_2 & \beta_3 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \beta_2 & \beta_3 \\
a & b\beta_2 & c\beta_3 & a\xi^* & b\xi^*\beta_2 & c\xi^*\beta_3 \\
r' & s'\beta_2 & t'\beta_3 & r'\xi & s'\xi\beta_2 & t'\xi\beta_3
\end{pmatrix}$$

has determinant $\beta_2^2\beta_3^2(\xi^* - \xi)(t - r - s)(t' - r' - s') \neq 0$, then the correct value for v_i is unpredictable: an invalid ciphertext will be accepted with probability $1/p$.

2. Let us now consider the mask $u_1^{z_1} u_2^{z_2} u_3^{z_3}$: its discrete logarithm in basis g_1 is $rz_1 + s\beta_2 z_2 + t\beta_3 z_3$, whereas the informations about (z_1, z_2, z_3) are $h_1 = g_1^{z_1} g_3^{z_3}$ and $h_2 = g_2^{z_2} g_3^{z_3}$. The matrix

$$\begin{pmatrix}
1 & 0 & \beta_3 \\
0 & \beta_2 & \beta_3 \\
r & s\beta_2 & t\beta_3
\end{pmatrix}$$

has determinant $\beta_2\beta_3(t - r - s)(t' - r' - s') \neq 0$, then the value of the mask is unpredictable: in this game, e_1^* is perfectly uniformly distributed in $\mathbb{G}$.

Since the unique difference between the two games is the linear/random tuple, unless a collision is found for $\mathfrak{H}_K$ (probability bounded by $\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t)$) and or an invalid ciphertext is accepted (probability bounded by q_d/p), then

$$\Pr_2[1 \leftarrow \mathcal{B}] \geq \Pr_1[1 \leftarrow \mathcal{B}] - \mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}.$$

– In game $\mathcal{G}_3$, to encrypt the challenge vectors $\boldsymbol{M}_b$ and $\boldsymbol{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}_1^*$: for a random $t_1^* \xleftarrow{\$} \mathbb{Z}_p$, $\mathbf{u}_1^* = (g_1^{r_1^*}, g_2^{s_1^*}, g_3^{t_1^*})$, $e_1^* \xleftarrow{\$} \mathbb{G}$, and $v_1^* \xleftarrow{\$} \mathbb{G}$. As just explained, this is perfectly indistinguishable with the previous game:

$$\Pr_3[1 \leftarrow \mathcal{B}] = \Pr_2[1 \leftarrow \mathcal{B}] \geq (\mathsf{Adv}^{\mathsf{ind-pd-cca}}_{n-\mathsf{DLCS}}(\mathcal{A}) - 1)/2 - \mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}.$$

– In game $\mathcal{G}_4$, to encrypt the challenge vectors $\boldsymbol{M}_b$ and $\boldsymbol{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}^*$: for a random vector $\boldsymbol{t}^* \xleftarrow{\$} \mathbb{Z}_p^n$, for $i = 2, \ldots, n$: $\mathbf{u}_i^* = (g_1^{r_i^*}, g_2^{s_i^*}, g_3^{t_i^*})$, $e_i^* \xleftarrow{\$} \mathbb{G}$, and $v_i^* \xleftarrow{\$} \mathbb{G}$. Thus replacing sequentially the $\mathcal{C}_i^*$'s by random ones, as we've just done, we obtain

$$\Pr_4[1 \leftarrow \mathcal{B}] \leq \Pr_3[1 \leftarrow \mathcal{B}] - (n-1)\left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}\right).$$

– In game $\mathcal{G}_5$, to encrypt the challenge vectors $\boldsymbol{M}_b$ and $\boldsymbol{N}_b$, $\mathcal{B}$ does as above, except for $\mathcal{C}'^*$: for a random vector $\boldsymbol{c}^* \xleftarrow{\$} \mathbb{Z}_p^n$, for $i = 1, \ldots, n$: $\boldsymbol{\alpha}_i^* = (g_1^{a_i^*}, g_2^{b_i^*}, g_3^{c_i^*})$, $\beta_i^* \xleftarrow{\$} \mathbb{G}$, and $\gamma_i^* \xleftarrow{\$} \mathbb{G}$. Thus replacing sequentially the $\mathcal{C}_i'^*$'s by random ones, as we've just done, we obtain

$$\Pr_5[1 \leftarrow \mathcal{B}] \leq \Pr_4[1 \leftarrow \mathcal{B}] - n\left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}\right).$$

In this last game, it is clear that $\Pr_5[1 \leftarrow \mathcal{B}] = 1/2$, since $(\boldsymbol{M}_b, \boldsymbol{N}_b)$ is not used anymore:

$$\frac{\mathsf{Adv}^{\mathsf{ind-pd-cca}}_{n-\mathsf{DLCS}}(\mathcal{A}) - 1}{2} - 2n \times \left(\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t) - \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) - \frac{q_d}{p}\right) \leq \frac{1}{2},$$

which concludes the proof. □

B.3 Double Linear Cramer-Shoup (DLCS) Commitment

Recently, Lindell [Lin11] proposed a highly efficient UC commitment. Our commitment strongly relies on it, but does not need to be UC secure. We will then show that the decommitment check can be done in an implicit way with an appropriate smooth projective hash function. Basically, the technique consists in encrypting M in $\mathcal{C} = (\mathbf{u}, e, v) = \mathsf{LCS}(\ell, M; r, s)$, also getting $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$, and then encrypting $1_{\mathbb{G}}$ in $\mathcal{C}' = \mathsf{LCS}^*(\ell, 1_{\mathbb{G}}, \xi; a, b)$, with the same ξ. For a given challenge ε, we can see $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon} = \mathsf{LCS}^*(\ell, M, \xi; r + \varepsilon a, s + \varepsilon b)$, where the computations are done component-wise, as an encryption of M, still using the same above ξ. Note that Lindell [Lin11] used $\mathcal{C}^{\varepsilon} \cdot \mathcal{C}'$, but our choice seems more natural, since we essentially re-randomize the initial encryption $\mathcal{C}$, but we have to take care of choosing $\varepsilon \neq 0$. It makes use of an equivocable commitment: the Pedersen commitment [Ped92].

– $\mathsf{Setup}(1^k)$ generates a group $\mathbb{G}$ of order p, with two independent generators g and ζ;
– $\mathsf{Commit}(m; r)$, for a message $m \xleftarrow{\$} \mathbb{Z}_p$ and random coins $r \xleftarrow{\$} \mathbb{Z}_p$, produces a commitment $c = g^m \zeta^r$;
– $\mathsf{Decommit}(c, m; r)$ outputs m and r, which opens c into m, with checking ability: $c \overset{?}{=} g^m \zeta^r$.

This commitment is computationally binding under the discrete logarithm assumption: two different openings (m, r) and (m', r') for a commitment c, leads to the discrete logarithm of ζ in basis g, that is equal to $(m' - m) \cdot (r - r')^{-1} \bmod p$. Granted this logarithm as additional information from the setup, one can equivocate any dummy commitment.

Description. Our n-message vector commitment, which includes labels, is depicted on Figure 6, where the computation between vectors are component-wise. We assume we commit vectors of group elements, but they can come from the reversible transformation $\mathcal{G}$. Note that for this commitment scheme, we can use $\boldsymbol{\varepsilon} = (\varepsilon, \dots, \varepsilon)$. For the version with SPHF implicit verification, according to the language, one can have to use independent components $\boldsymbol{\varepsilon} \overset{\$}{\leftarrow} (\mathbb{Z}_p^*)^n$.

- $\mathsf{Setup}(1^k)$: A group $\mathbb{G}$ of prime order p, with ten independent generators $(g_1, g_2, g_3, h_1, h_2, c_1, c_2, d_1, d_2, \zeta) \overset{\$}{\leftarrow} \mathbb{G}^{10}$, a collision-resistant hash function $\mathfrak{H}_K$, and possibly an additional reversible mapping $\mathcal{G}$ from $\{0,1\}^k$ to $\mathbb{G}$ to commit to bit-strings. One can denote $\mathsf{ek} = (c_1, c_2, d_1, d_1, h_1, h_2, \mathfrak{H}_K)$;
- $\mathsf{Commit}(\ell, \boldsymbol{M}; \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{b}, t)$: for $(\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{b}, t) \overset{\$}{\leftarrow} \mathbb{Z}_p^{4n+1}$
  - $(\mathcal{C}, \mathcal{C}') \leftarrow n - \mathsf{DLCS}(\ell, \mathsf{ek}, \boldsymbol{M}, (1_{\mathbb{G}})^n; \boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{b})$
  - $\chi = \mathfrak{H}_K(\boldsymbol{M}, \mathcal{C}'), \mathcal{C}'' = g_1^t \zeta^\chi$ $\quad \xrightarrow{\mathcal{C}, \mathcal{C}''}$
  - $\prod_i \varepsilon_i \overset{?}{\neq} 0 \bmod p$ $\quad \xleftarrow{\boldsymbol{\varepsilon}}$ $\quad \varepsilon \overset{\$}{\leftarrow} \mathbb{Z}_p^*, \boldsymbol{\varepsilon} \leftarrow (\varepsilon, \dots, \varepsilon)$
  - $\boldsymbol{z} = (\boldsymbol{r} + \boldsymbol{\varepsilon} \cdot \boldsymbol{a} \bmod p, \boldsymbol{s} + \boldsymbol{\varepsilon} \cdot \boldsymbol{b} \bmod p)$
  - $\textsc{Erase}(\boldsymbol{r}, \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{b})$
- $\mathsf{Decommit}(\ell, \mathcal{C}, \mathcal{C}', \boldsymbol{\varepsilon})$: $\quad \xrightarrow{\mathcal{C}', t, \boldsymbol{M}, \mathbf{z}}$ $\quad$ compute ξ from $\mathcal{C}$
  - $\chi = \mathfrak{H}_K(\boldsymbol{M}, \mathcal{C}'), \mathcal{C}'' \overset{?}{=} g_1^t \zeta^\chi$
  - $\mathcal{C} \cdot \mathcal{C}'^{\boldsymbol{\varepsilon}} \overset{?}{=} n - \mathsf{LCS}^*(\ell, \boldsymbol{M}, \xi; \mathbf{z}_r, \mathbf{z}_s)$

Fig. 6. $n - \mathsf{DLCS}$ Commitment Scheme

Analysis. Let us briefly show the properties of this commitment:

- Hiding property: $\boldsymbol{M}$ is committed in the Pedersen commitment $\mathcal{C}''$, that does not leak any information, and in the $n - \mathsf{LCS}$ encryption $\mathcal{C}$, that is indistinguishable, even with access to the decryption oracle (extractability). This also implies non-malleability.
- Binding property: $\boldsymbol{M}$, after having been hashed, is committed in the Pedersen commitment $\mathcal{C}''$, that is computationally binding.
- Extractability: using the decryption key of the LCS encryption scheme, one can extract $\boldsymbol{M}$ from $\mathcal{C}$. Later, one has to open the ciphertext $\mathcal{C}\mathcal{C}'^{\boldsymbol{\varepsilon}}$ with $\boldsymbol{M}'$, which can be different from $\boldsymbol{M}$ in the case that $\mathcal{C}'$ contains $\boldsymbol{N} \neq (1_{\mathbb{G}})^n$. But then $\boldsymbol{M}' = \boldsymbol{M} \cdot \boldsymbol{N}^{\boldsymbol{\varepsilon}}$, that is unpredictable at the commit time of $\mathcal{C}''$. With probability at most $1/p$, one can open the commitment with a value $\boldsymbol{M}'$ different from $\boldsymbol{M}$, if this value $\boldsymbol{M}'$ has been correctly anticipated in $\mathcal{C}''$.
- Equivocability: if one wants to open with $\boldsymbol{M}'$, one can compute $\boldsymbol{N} = (\boldsymbol{M}'/\boldsymbol{M})^{1/\boldsymbol{\varepsilon}}$, encrypt $\boldsymbol{N}$ in $\mathcal{C}' = n - \mathsf{LCS}^*(\ell, \boldsymbol{N}, \xi; \boldsymbol{a}, \boldsymbol{b})$, and update χ and t, using the Pedersen trapdoor for equivocability.

To allow an implicit verification with SPHF, one omits to send $\boldsymbol{M}$ and $\mathbf{z}$, but make an implicit proof of their existence. Therefore, $\boldsymbol{M}$ cannot be committed/verified in $\mathcal{C}''$, which has an impact on the binding property: $\mathcal{C}$ and $\mathcal{C}''$ are not binded to a specific $\boldsymbol{M}$, even in a computational way. However, as said above, if $\mathcal{C}''$ contains a ciphertext $\mathcal{C}'$ of $\boldsymbol{N} \neq (1_{\mathbb{G}})^n$, the actual committed value will depend on $\boldsymbol{\varepsilon}$: $\boldsymbol{M}' = \boldsymbol{M}\boldsymbol{N}^{\boldsymbol{\varepsilon}}$ has its i-component, where $N_i \neq 1_{\mathbb{G}}$, uniformly distributed in $\mathbb{G}$ when ε is uniformly distributed in $\mathbb{Z}_p^*$. In addition, if $\boldsymbol{\varepsilon} \overset{\$}{\leftarrow} (\mathbb{Z}_p^*)^n$, all these i-component where $N_i \neq 1_{\mathbb{G}}$ are randomly and independently distributed in $\mathbb{G}$. Then, if the committed value has to satisfy a specific relation, with very few solutions, $\boldsymbol{M}'$ will unlikely satisfy it.

C Smooth Projective Hash Functions on More Complex Languages

C.1 Basic Relations

We first consider Diffie-Hellman pairs and linear tuples and show we can make proof of membership without using any pairing.

DDH pairs. Let us assume a user is given two elements g, h and then wants to send $G = g^a, H = h^a$ for a chosen a and prove that the pair (G, H) is well-formed with respect to (g, h). We thus consider the language of Diffie Hellman tuples $(g, h, G = g^a, H = h^a)$, with a as a witness.

As done in [CS98], we define a projection key $\mathsf{hp} = g^{x_1}h^{x_2}$ by picking two random scalars $x_1, x_2 \overset{\$}{\leftarrow} \mathbb{Z}_p$, which define the secret hashing key $\mathsf{hk} = (x_1, x_2)$. One can then compute the hash value in two different ways: $\mathsf{ProjHash}(\mathsf{hp}, (g, h, G, H), a) \overset{\text{def}}{=} \mathsf{hp}^a = (g^{ax_1}h^{ax_2}) = G^{x_1}H^{x_2} \overset{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, (g, h, G, H))$.

Such SPHF is smooth: this can be seen by proceeding like in the Cramer-Shoup proof. Given $\mathsf{hp} = g^\alpha$, $h = g^\beta$, $G = g^a$ and $H = h^{a'}$, the hash value is g^γ that satisfies:

$$\begin{pmatrix}\alpha\\ \gamma\end{pmatrix} = \begin{pmatrix}1 & \beta\\ a & \beta a'\end{pmatrix} \cdot \begin{pmatrix}x_1\\ x_2\end{pmatrix}$$

The determinant of this matrix is $\Delta = \beta(a' - a)$, that is zero if and only if we do have a valid Diffie-Hellman tuple. Otherwise, from hp, γ is perfectly hidden, from an information theoretical point of view, and so is $\mathsf{Hash}(\mathsf{hk}, (g, h, G, H))$ too.

DLin tuples. Let us consider three generators u, v, w, and a tuple $U = u^r, V = v^s, W = w^t$ one wants to prove be linear (*i.e.* $t = r + s$). We first define two projection keys $\mathsf{hp}_1 = u^{x_1}w^{x_3}, \mathsf{hp}_2 = v^{x_2}w^{x_3}$, for random scalars that define the secret hashing key $\mathsf{hk} = (x_1, x_2, x_3)$. One can then compute the hash value in two different ways: $\mathsf{ProjHash}(\mathsf{hp}_1, \mathsf{hp}_2, (u, v, w, U, V, W), r, s) \overset{\text{def}}{=} \mathsf{hp}_1^r\mathsf{hp}_2^s = (u^{rx_1}v^{sx_2}w^{x_3(r+s)}) = U^{x_1}V^{x_2}W^{x_3} \overset{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, (u, v, w, U, V, W))$.

Once again this SPHF can be shown to be smooth: given $\mathsf{hp}_1 = u^\alpha$, $\mathsf{hp}_2 = u^\beta$, $v = u^\gamma$, $w = u^\delta$, the hash value is u^λ that satisfies:

$$\begin{pmatrix}\alpha\\ \beta\\ \lambda\end{pmatrix} = \begin{pmatrix}1 & 0 & \delta\\ 0 & \gamma & \delta\\ r & \gamma s & \delta t\end{pmatrix} \cdot \begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}$$

The determinant of this matrix is $\Delta = \gamma\delta(t - s - r)$, that is zero if and only if we do have a valid linear tuple.

C.2 Smooth Projective Hashing on Commitments

We now show that our commitments LCS or DLCS' are well-suited for a use together with smooth projective hash functions: instead of publishing $\boldsymbol{z}$ at the decommit phase, in order to check whether $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon} \overset{?}{=} \mathsf{LCS}^*(\ell, M, \xi; z_r, z_s)$ (with $\varepsilon = 0$ in the LCS non-equivocable case, or with $\varepsilon \neq 0$ in the DLCS' case), one uses a smooth projective hash function to "implicitly" prove the existence of a witness that the commitment actually contains the claimed (or assumed) value M. We will thereafter be able to use this primitive in Language-Authenticated Key Exchange, for complex languages.

Smooth projective hash functions. We thus have a commitment, either $\mathcal{C}$ or $\mathcal{C} \cdot \mathcal{C}'^{\varepsilon}$, but we use in both cases the notation $\mathcal{C}$, and want to check whether there exists $\mathbf{z} = (z_r, z_s)$ such that

$$\mathcal{C} = \mathsf{LCS}^*(\ell, M, \xi; z_r, z_s) = (\mathbf{u} = (g_1^{z_r}, g_2^{z_s}, g_3^{z_r+z_s}), e = M \cdot h_1^{z_r} h_2^{z_s}, v = v_1^{z_r} v_2^{z_s}),$$

where we denote $v_1 = c_1 d_1^{\xi}$ and $v_2 = c_2 d_2^{\xi}$. We note here that all the bases g_1, g_2, g_3, h_1, h_2 but also v_1, v_2 are known as soon as ξ is known (the $\mathcal{C}$ part of the DLCS' commitment). One then generates $\mathsf{hk} = (\eta, \theta, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, and derives the projection key that depends on ξ only: $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta} g_3^{\kappa} h_1^{\lambda} v_1^{\mu}, \mathsf{hp}_2 = g_2^{\theta} g_3^{\kappa} h_2^{\lambda} v_2^{\mu})$. Then, one can compute the hash value:

$$H = \mathsf{Hash}(\mathsf{hk}, M, \mathcal{C}) \stackrel{\mathsf{def}}{=} u_1^{\eta} u_2^{\theta} u_3^{\kappa} (e/M)^{\lambda} v^{\mu} = \mathsf{hp}_1^{z_r} \mathsf{hp}_2^{z_s} \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, \mathcal{C}; z_r, z_s) = H'.$$

Security properties. Let us claim and prove the security properties:

Theorem 8. *Under the* DLin *assumption, the above smooth projective hash function is both smooth and pseudo-random:*

- *Smoothness:* $\mathsf{Adv}_{\Pi}^{\mathsf{smooth}} = 0$*;*
- *Pseudo-Randomness:* $\mathsf{Adv}_{\Pi}^{\mathsf{pr}}(t) \le \mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{dlin}}(t)$*.*

Proof. For the correctness, one can easily check that if $\mathcal{C}$ contains $M = M'$, then $H = H'$:

$$\begin{aligned} H &= u_1^{\eta} u_2^{\theta} u_3^{\kappa} (e/M)^{\lambda} v^{\mu} = (g_1^{z_r})^{\eta} (g_2^{z_s})^{\theta} (g_3^{z_r+z_s})^{\kappa} (h_1^{z_r} h_2^{z_s} M'/M)^{\lambda} (v_1^{z_r} v_2^{z_s})^{\mu} \\ &= (g_1^{\eta} g_3^{\kappa} h_1^{\lambda} v_1^{\mu})^{z_r} \cdot (g_2^{\theta} g_3^{\kappa} h_2^{\lambda} v_2^{\mu})^{z_s} \cdot (M'/M)^{\lambda} = \mathsf{hp}_1^{z_r} \mathsf{hp}_2^{z_s} \cdot (M'/M)^{\lambda} = H' \cdot (M/M')^{\lambda} \end{aligned}$$

Smoothness: if $\mathcal{C}$ is not a correct encryption of M, then H is unpredictable: let us denote M' and z'_s such that $\mathcal{C} = (\boldsymbol{u} = (g_1^{z_r}, g_2^{z_s}, g_3^{z_t}), e = M' h_1^{z_r} h_2^{z_s}, v = v_1^{z_r} v_2^{z'_s})$. Then, if we denote $g_2 = g_1^{\beta_2}$ and $g_3 = g_1^{\beta_3}$, and $h_1 = g_1^{\rho_1}$ and $h_2 = g_1^{\rho_2}$, but also $v_1 = g_1^{\delta_1}$ and $v_2 = g_1^{\delta_2}$, and $\Delta = \log_{g_1}(M'/M)$:

$$\begin{aligned} H &= g_1^{\eta z_r} g_1^{\beta_2 \theta z_s} g_1^{\beta_3 \kappa z_t} (M'/M)^{\lambda} (g_1^{\rho_1 z_r + \rho_2 z_s})^{\lambda} (v_1^{z_r} v_2^{z'_s})^{\mu} \\ \log_{g_1} H &= \eta z_r + \beta_2 \theta z_s + \beta_3 \kappa z_t + \lambda(\rho_1 z_r + \rho_2 z_s) + \mu(\delta_1 z_r + \delta_2 z'_s) + \lambda \Delta \end{aligned}$$

The information leaked by the projected key is $\log_{g_1} \mathsf{hp}_1 = \eta + \beta_3 \kappa + \rho_1 \lambda + \delta_1 \mu$ and $\log_{g_1} \mathsf{hp}_2 = \beta_2 \theta + \beta_3 \kappa + \rho_2 \lambda + \delta_2 \mu$, which leads to the matrix

$$\begin{pmatrix} 1 & 0 & \beta_3 & \rho_1 & \delta_1 \\ 0 & \beta_2 & \beta_3 & \rho_2 & \delta_2 \\ z_r & \beta_2 z_s & \beta_3 z_t & \Delta + \rho_1 z_r + \rho_2 z_s & \delta_1 z_r + \delta_2 z'_s \end{pmatrix}$$

One remarks that if $z_t \neq z_r + z_s \bmod p$, then the three rows are not linearly dependent even considering the 3 first components only, and then H is unpredictable. Hence, we can assume that $z_t = z_r + z_s \bmod p$. The third row must thus be the first multiplied by z_r plus the second multiplied by z_s: $\rho_2 z_s = \Delta + \rho_2 z_s \bmod p$ and $z_s = z'_s \bmod p$, which implies $z'_s = s$ and $\Delta = 0$, otherwise, H remains unpredictable.

As a consequence, if $\mathcal{C}$ is not a correct encryption of W, H is perfectly unpredictable in $\mathbb{G}$:

$$\begin{aligned} &\{(\mathsf{hp}, H), \mathsf{hk} = (\eta, \theta, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5, \mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta} g_3^{\kappa} h_1^{\lambda} v_1^{\mu}, \mathsf{hp}_2 = g_2^{\theta} g_3^{\kappa} h_2^{\lambda} v_2^{\mu}), H \leftarrow \mathsf{Hash}(\mathsf{hk}, M, \mathcal{C})\} \\ &\approx_s \{(\mathsf{hp}, H), \mathsf{hk} = (\eta, \theta, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5, \mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta} g_3^{\kappa} h_1^{\lambda} v_1^{\mu}, \mathsf{hp}_2 = g_2^{\theta} g_3^{\kappa} h_2^{\lambda} v_2^{\mu}), H \xleftarrow{\$} \mathbb{G}\}. \end{aligned}$$

Pseudo-Randomness: we've just shown that if $\mathcal{C}$ is not a correct encryption of M, then H is statistically unpredictable. Let us be given a triple (g_1, g_2, g_3) together with another triple $\boldsymbol{u} = (u_1 = g_1^a, u_2 = g_2^b, u_3 = g_3^c)$. We choose random exponents $(x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3)$, and for $i = 1, 2$, we set $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. We generate $\mathcal{C} = (\boldsymbol{u}, e = M \cdot u_1^{z_1} u_2^{z_2} u_3^{z_3}, v = u_1^{x_1+\xi y_1} u_2^{x_2+\xi y_2} u_3^{x_3+\xi y_3})$. If $c = a + b \bmod p$ (i.e., $\boldsymbol{u}$ is a linear tuple in basis $\boldsymbol{g}$), then $\mathcal{C}$ is a valid encryption of M, otherwise this is not, and we can apply the smoothness property:

$$\mathsf{Adv}_{\Pi}^{\mathsf{pr}}(t) \leq \mathsf{Adv}_{\Pi}^{\mathsf{smooth}} + \mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{dlin}}(t) \leq \mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{dlin}}(t).$$

□

C.3 Single Equation

Let us assume that we have $\mathcal{Y}_i$ committed in $\mathbb{G}$, in $\boldsymbol{c}_i$, for $i = 1, \ldots, m$ and $\mathcal{Z}_i$ committed in $\mathbb{G}_T$, in $\boldsymbol{D}_i$, for $i = m+1, \ldots, n$, and we want to show they simultaneously satisfy

$$\left(\prod_{i=1}^{m} e(\mathcal{Y}_i, \mathcal{A}_i)\right) \cdot \left(\prod_{i=m+1}^{n} \mathcal{Z}_i^{\mathfrak{z}_i}\right) = \mathcal{B}$$

where $\mathcal{A}_i \in \mathbb{G}$, $\mathcal{B} \in \mathbb{G}_T$, and $\mathfrak{z}_i \in \mathbb{Z}_p$ are public. As already said, the commitment can either be the LCS or the DLCS' version, but they both come up to a ciphertext $\mathcal{C}$ with the appropriate random coins $\mathbf{z}$:

$\boldsymbol{c}_i = (\boldsymbol{u_i} = (g_1^{z_{r_i}}, g_2^{z_{s_i}}, g_3^{z_{r_i}+z_{s_i}}), e_i = h_1^{z_{r_i}} h_2^{z_{s_i}} \cdot \mathcal{Y}_i, v_i = (c_1 d_1^{\xi})^{z_{r_i}} \cdot (c_2 d_2^{\xi})^{z_{s_i}})$
for i=1,...,m
which can be transposed into $\mathbb{G}_T$:
$\boldsymbol{C}_i = (\boldsymbol{U_i} = (G_{i,1}^{z_{r_i}}, G_{i,2}^{z_{s_i}}, G_{i,3}^{z_{r_i}+z_{s_i}}), E_i = H_{i,1}^{z_{r_i}} H_{i,2}^{z_{s_i}} \cdot \mathcal{Z}_i, V_i = (C_{i,1} D_{i,1}^{\xi})^{z_{r_i}} \cdot (C_{i,2} D_{i,2}^{\xi})^{z_{s_i}})$
for i=1,...,m
where, for $j = 1, 2, 3$, $G_{i,j} = e(g_j, \mathcal{A}_i)$ and for $j = 1, 2$, $H_{i,j} = e(h_j, \mathcal{A}_i)$, $C_{i,j} = e(c_j, \mathcal{A}_i)$, $D_{i,j} = e(d_j, \mathcal{A}_i)$, but also, $\mathcal{Z}_i = e(\mathcal{Y}_i, \mathcal{A}_i)$, and
$\boldsymbol{D}_i = (\boldsymbol{U_i} = (G_{i,1}^{z_{r_i}}, G_{i,2}^{z_{s_i}}, G_{i,3}^{z_{r_i}+z_{s_i}}), E_i = H_{i,1}^{z_{r_i}} H_{i,2}^{z_{s_i}} \cdot \mathcal{Z}_i, V_i = (C_{i,1} D_{i,1}^{\xi})^{z_{r_i}} \cdot (C_{i,2} D_{i,2}^{\xi})^{z_{s_i}})$
for $i = m+1, \ldots, n$ where, for $j = 1, 2, 3$, $G_{i,j} = e(g_j, g)$
and for $j = 1, 2$, $H_{i,j} = e(h_j, g)$, $C_{i,j} = e(c_j, g)$, $D_{i,j} = e(d_j, g)$

where g is a generator of $\mathbb{G}$ and $\xi = \mathfrak{H}_K(\boldsymbol{u_1}, \ldots, \boldsymbol{u_m}, \boldsymbol{U_{m+1}}, \ldots, \boldsymbol{U_n}, e_1, \ldots, e_m, E_{m+1}, \ldots, E_n)$: $\mathbb{G}$-elements are encrypted under $\mathsf{ek} = (\boldsymbol{g} = (g_1, g_2, g_3), \boldsymbol{h} = (h_1, h_2), \boldsymbol{c} = (c_1, d_1), \boldsymbol{d} = (c_2, d_2))$, and $\mathbb{G}_T$-element are encrypted under $\mathsf{EK}_i = (\boldsymbol{G_i} = (G_{i,1}, G_{i,2}, G_{i,3}), \boldsymbol{H}_i = (H_{i,1}, H_{i,2}), \boldsymbol{C}_i = (C_{i,1}, C_{i,2}), \boldsymbol{D}_i = (D_{i,1}, D_{i,2}))$. Note that an additional label ℓ can be included in the computation of ξ.

For the hashing keys, one picks scalars $(\lambda, (\eta_i, \theta_i, \kappa_i, \mu_i)_{i=1,\ldots,n}) \overset{\$}{\leftarrow} \mathbb{Z}_p^{4n+1}$, and sets $\mathsf{hk}_i = (\eta_i, \theta_i, \kappa_i, \lambda, \mu_i)$.

One then computes the projection keys as $\mathsf{hp}_i = (g_1^{\eta_i} g_3^{\kappa_i} h_1^{\lambda} (c_1 d_1^{\xi})^{\mu_i}, g_2^{\theta_i} g_3^{\kappa_i} h_2^{\lambda} (c_2 d_2^{\xi})^{\mu_i}) \in \mathbb{G}^2$. The associated projection keys in $\mathbb{G}_T$ are $\mathsf{HP}_i = (e(\mathsf{hp}_{i,1}, \mathcal{A}_i), e(\mathsf{hp}_{i,2}, \mathcal{A}_i))$, for $i = 1, \ldots, n$, where $\mathcal{A}_i = g^{\mathfrak{z}_i}$ for $i = m+1, \ldots, n$.

The hash value is

$$\begin{aligned}
H &= \left(\prod_i U_{i,1}^{\eta_i} \cdot U_{i,2}^{\theta_i} \cdot U_{i,3}^{\kappa_i} \cdot E_i^{\lambda} \cdot V_i^{\mu_i}\right) \cdot \mathcal{B}^{-\lambda} \\
&= \left(\prod_i \mathsf{HP}_{i,1}^{z_{r_i}} \mathsf{HP}_{i,2}^{z_{s_i}} \mathcal{Z}_i^{\lambda}\right) \cdot \mathcal{B}^{-\lambda} = \left(\prod_i \mathsf{HP}_{i,1}^{z_{r_i}} \mathsf{HP}_{i,2}^{z_{s_i}}\right) \left(\left(\prod_{i=1}^{m} e(\mathcal{Y}_i, \mathcal{A}_i)\right) \left(\prod_{i=m+1}^{n} \mathcal{Z}_i\right) / \mathcal{B}\right)^{\lambda} \\
&= \prod_i \mathsf{HP}_{i,1}^{z_{r_i}} \mathsf{HP}_{i,2}^{z_{s_i}} = \left(\prod_{i=1}^{m} e(\mathsf{hp}_{i,1}^{z_{r_i}}, \mathcal{A}_i) \cdot e(\mathsf{hp}_{i,2}^{z_{s_i}}, \mathcal{A}_i)\right) \cdot \left(e(\prod_{i=m+1}^{n} \mathsf{hp}_{i,1}^{z_{r_i}}, g^{\mathfrak{z}_i}) \cdot e(\prod_{i=m+1}^{n} \mathsf{hp}_{i,2}^{z_{s_i}}, g^{\mathfrak{z}_i})\right)
\end{aligned}$$

which can be computed either from the commitments and the hashing keys, or from the projection keys and the witnesses. We prove below the smoothness, but first extend it even more to several equations.

C.4 Multiple Equations

Let us assume that we have $\mathcal{Y}_i$ committed in $\mathbb{G}$, in $\boldsymbol{c}_i$, for $i = 1, \ldots, m$ and $\mathcal{Z}_i$ committed in $\mathbb{G}_T$, in $\boldsymbol{D}_i$, for $i = m+1, \ldots, n$, and we want to show they simultaneously satisfy

$$\left(\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i})\right) \cdot \left(\prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}}\right) = \mathcal{B}_k, \text{ for } k = 1, \ldots, t.$$

where $\mathcal{A}_{k,i} \in \mathbb{G}$, $\mathcal{B}_k \in \mathbb{G}_T$, and $\mathfrak{z}_{k,i} \in \mathbb{Z}_p$, as well as $A_k \subseteq \{1, \ldots, m\}$ and $B_k \subseteq \{m+1, \ldots, n\}$ are public. As above, from the commitments, one derives the global ξ, which can also involves the label ℓ, and one can also derive the commitments in $\mathbb{G}_T$, $\boldsymbol{C}_{k,i}$ that correspond to the encryption of $\mathcal{Z}_{k,i} = e(\mathcal{Y}_i, \mathcal{A}_{k,i})$ under the keys $\mathsf{EK}_{k,i} = (\boldsymbol{G}_{k,i} = (G_{k,i,1}, G_{k,i,2}, G_{k,i,3}), \boldsymbol{H}_{k,i} = (H_{k,i,1}, H_{k,i,2}), \boldsymbol{C}_{k,i} = (C_{k,i,1}, C_{k,i,2}), \boldsymbol{D}_{k,i} = (D_{k,i,1}, D_{k,i,2}))$, where the capital letters $X_{k,i,j}$ correspond to the lower-case letters x_j paired with $\mathcal{A}_{k,i}$.

For the hashing keys, one picks scalars $(\lambda, \{\eta_i, \theta_i, \kappa_i, \mu_i\}_{i=1,\ldots,n}) \xleftarrow{\$} \mathbb{Z}_p^{4n+1}$, $\{\varepsilon_k\}_{k=1,\ldots,t} \xleftarrow{\$} \mathbb{Z}_p^t$ and sets $\mathsf{hk} = (\{\mathsf{hk}_i = (\eta_i, \theta_i, \kappa_i, \lambda, \mu_i)\}_{i=1,\ldots,n}, \{\varepsilon_k\}_{k=1,\ldots,t})$. We insist on the fact that the ε_k's have to be sent after the commitments have been sent, or at least committed to (such as $\mathcal{C}$ and $\mathcal{C}''$ which prevent from any modification). One then computes the projection keys as $\mathsf{hp}_i = (g_1^{\eta_i} g_3^{\kappa_i} h_1^{\lambda} (c_1 d_1^{\xi})^{\mu_i}, g_2^{\theta_i} g_3^{\kappa_i} h_2^{\lambda} (c_2 d_2^{\xi})^{\mu_i}) \in \mathbb{G}^2$, together with ε_k. The associated projection keys in $\mathbb{G}_T$ are $\mathsf{HP}_{k,i} = (e(\mathsf{hp}_{i,1}, \mathcal{A}_{k,i}), e(\mathsf{hp}_{i,2}, \mathcal{A}_{k,i}))$, for $k = 1, \ldots, t$ and $i = 1, \ldots, n$, where $\mathcal{A}_{k,i} = g^{\mathfrak{z}_{k,i}}$ for $i = m+1, \ldots, n$, together with ε_k. The hash function and the projective hash function are defined as:

$$\begin{aligned}
H &= \prod_k \left(\left(\prod_{i \in A_k \cup B_k} U_{k,i,1}^{\eta_i} \cdot U_{k,i,2}^{\theta_i} \cdot U_{k,i,3}^{\kappa_i} \cdot E_{k,i}^{\lambda} \cdot V_{k,i}^{\mu_i} \right) \cdot \mathcal{B}_k^{-\lambda} \right)^{\varepsilon_k} \\
&= \prod_k \left(\prod_{i \in A_k \cup B_k} \mathsf{HP}_{k,i,1}^{z_{r_i}} \cdot \mathsf{HP}_{k,i,2}^{z_{s_i}} \right)^{\varepsilon_k} \cdot \prod_k \left(\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}} \cdot \mathcal{B}_k^{-1} \right)^{\lambda \varepsilon_k} \\
H' &= \prod_k \left(\prod_{i \in A_k \cup B_k} \mathsf{HP}_{k,i,1}^{z_{r_i}} \cdot \mathsf{HP}_{k,i,2}^{z_{s_i}} \right)^{\varepsilon_k}
\end{aligned}$$

which can be computed either from the commitments and the hashing keys, or from the projection keys and the witnesses. They lead to the same values $H' = H$ if

- for every k, $\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}} = \mathcal{B}_k$, which means that all the equations are simultaneously satisfied;
- $\lambda = 0$, which is quite unlikely;
- $\prod_k \Delta_k^{\varepsilon_k} = 1$, where for every k, $\Delta_k = \prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}} / \mathcal{B}_k$, which is also quite unlikely since the Δ_k's are fixed before the ε_k's are known.

C.5 Security Analysis

Smoothness. In this section, first we prove the smoothness of the SPHF built right before. For $k = 1$, this proves the smoothness of the SPHF built to handle variables in one linear pairing equation. The list of commitments $\mathcal{C} = (\mathcal{C}_1, \ldots, \mathcal{C}_n)$, which possibly results from the multiplication by the companion ciphertext when using the equivocable variant, should be considered in the language if and only if:

- the commitments are all valid Linear Cramer-Shoup ciphertexts (in either $\mathbb{G}$ or $\mathbb{G}_T$), with the common and fixed ξ;
- the plaintexts satisfy the linear pairing product equations.

Let us assume that one of the commitments is not a valid ciphertext, this means that for some index $i \in \{1, \dots, n\}$, the ciphertext $(\boldsymbol{U_i} = (G_1^{r_i}, G_2^{s_i}, G_3^{t_i}), E_i, V_i)$ in $\mathbb{G}_T$ is such that either $t_i \neq r_i + s_i$ or $V_i \neq (C_1 D_1^{\xi})^{r_i} \cdot (C_2 D_2^{\xi})^{s_i}$. Then, the contribution of this ciphertext in the hash value is $(U_{i,1}^{\eta_i} \cdot U_{i,2}^{\theta_i} \cdot U_{i,3}^{\kappa_i} \cdot E_i^{\lambda} \cdot V_i^{\mu_i})^{\varepsilon'_i}$, where $\varepsilon'_i = \sum_{k, i \in A_k \cup B_k} \varepsilon_k$, knowing the projection keys that reveal, at most,

$$\log_{g_1} \mathsf{hp}_{i,1} = \eta_i + x_3 \cdot \kappa_i + x_4 \cdot \lambda + (y_1 + \xi y_3) \cdot \mu_i \quad \log_{g_1} \mathsf{hp}_{i,2} = x_2 \cdot \theta_i + x_3 \cdot \kappa_i + x_5 \cdot \lambda + (y_2 + \xi y_4) \cdot \mu_i,$$

where $g_2 = g_1^{x_2}$ $g_3 = g_1^{x_3}$ $h_1 = g_1^{x_4}$ $h_2 = g_1^{x_5}$ $c_1 = g_1^{y_1}$ $c_2 = g_1^{y_2}$ $d_1 = g_1^{y_3}$ $d_2 = g_1^{y_4}$. This contribution is thus $(G_1^{r_i \eta_i + x_2 s_i \theta_i + x_3 t_i \kappa_i + z_i \mu_i} \cdot E_i^{\lambda})^{\varepsilon'_i}$, where $V_i = G_1^{z_i}$. But even if all the discrete logarithms were known, and also λ, one has to guess $r_i \eta_i + x_2 s_i \theta_i + x_3 t_i \kappa_i + z_i \mu_i$, given $\eta_i + x_3 \cdot \kappa_i + (y_1 + \xi y_3) \cdot \mu_i$ and $x_2 \cdot \theta_i + x_3 \cdot \kappa_i + (y_2 + \xi y_4) \cdot \mu_i$:

$$\begin{pmatrix} 1 & 0 & x_3 & (y_1 + \xi y_3) \\ 0 & x_2 & x_3 & (y_2 + \xi y_4) \\ r_i & x_2 s_i & x_3 t_i & z_i \end{pmatrix}.$$

The first 3-column matrix has determinant is $x_2 x_3 (t_i - (r_i + s_i))$, that is non-zero as soon as $t_i \neq r_i + s_i$. In this case, there is no way to guess the correct value better than by chance: $1/p$. If $t_i = (r_i + s_i)$, the third line is linearly dependent with the 2 first, if and only if $z_i = r_i(y_1 + \xi y_3) + s_i(y_2 + \xi y_4)$. Otherwise, one has no better way to guess the value than by chance either. Hence the smoothness of this hash function when one commitment is not valid.

About the equation validity, the E_i's of the involved ciphertexts contain plaintexts $\mathcal{Y}_i$ or $\mathcal{Z}_i$, and contribute to the hash value: from the projection keys, the k-th equation contributes to

$$H_k = \left(\prod_{i \in A_k} \mathsf{HP}_{k,i,1}^{r_i} \cdot \mathsf{HP}_{k,i,2}^{s_i} \cdot \prod_{i \in B_k} \left(\mathsf{HP}_{i,1}^{r_i} \cdot \mathsf{HP}_{i,2}^{s_i} \right)^{\mathfrak{z}_{k,i}} \right)^{\varepsilon_k} \cdot \left(\prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}} \cdot \mathcal{B}_k^{-1} \right)^{\lambda \varepsilon_k}$$

Let us denote $\alpha_k = \prod_{i \in A_k} e(\mathcal{Y}_i, \mathcal{A}_{k,i}) \cdot \prod_{i \in B_k} \mathcal{Z}_i^{\mathfrak{z}_{k,i}} \cdot \mathcal{B}_k^{-1}$, then the uncertainty about H is $(\prod_k \alpha_k^{\varepsilon_k})^{\lambda}$. As soon as one of the equations is not satisfied, one of the α_k is different from 1. Since the ε_k's are unknown at the commitment time, one cannot make the α_k to compensate themselves, but by chance: if one equation is not satisfied, the probability that $\prod_k \alpha_k^{\varepsilon_k} = 1$ is $1/p$. Except this negligible case, $(\prod_k \alpha_k^{\varepsilon_k})^{\lambda}$ is totally unpredictable since λ is random.

Pseudo-randomness. The pseudo-randomness can be proven under the DLin assumption: with invalid ciphertexts, the smoothness guarantees unpredictability; without the witnesses, one cannot distinguish a valid ciphertext from an invalid ciphertext.

C.6 Asymmetric Setting

Our approach has been presented in the symmetric setting (at least when pairing are required). We can do the same in asymmetric bilinear groups, with $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and even more efficiently, using the Cramer-Shoup encryption scheme, and the analogous n-message commitment scheme, which security relies on the DDH assumption in either $\mathbb{G}_1$ or $\mathbb{G}_2$. In this setting, our methodology can handle linear pairing product equations:

$$\left(\prod_{i=1}^{m} e(\mathcal{X}_i, \mathcal{B}_i) \right) \cdot \left(\prod_{j=1}^{n} e(\mathcal{A}_j, \mathcal{Y}_j) \right) \cdot \left(\prod_{k=1}^{o} \mathcal{Z}_k^{\mathfrak{z}_k} \right) = g_T,$$

where $\mathcal{A}_j, \mathcal{B}_i, g_T$ are public values, in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively, and $\mathcal{X}_i, \mathcal{Y}_j, \mathcal{Z}_k$ are the unknown values, committed in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ respectively.

D Security of the LAKE Protocol: Proof of Theorem 1

For the sake of simplicity, we give in Figure 7 an explicit version of the protocol described in Figure 4. We omit the additional verification that all the committed values are in the correct subsets $\mathcal{P}$ and $\mathcal{S}$, since in the proof below we will always easily guarantee this membership. The proof heavily relies on the properties of the commitments and smooth projective hash functions given in Sections 3, 4 and Appendix B.

Initiator P_i — Receiver P_j

$(I0)$ $(\mathrm{VK}_i, \mathrm{SK}_i) \leftarrow \mathsf{KeyGen}()$
$\rightarrow \mathsf{pub}$
$\ell_i = (\ell, \mathsf{pub}, \mathrm{VK}_i, \mathrm{VK}_j)$

pre-flow $\xrightarrow{(\mathrm{VK}_i, \mathsf{pub}_i)}$ $\xleftarrow{(\mathsf{VK}_j, \mathsf{pub}_j)}$

$(R0)$ $(\mathsf{VK}_j, \mathsf{SK}_j) \leftarrow \mathsf{KeyGen}()$
$\rightarrow \mathsf{pub}$
$\ell_j = (\ell, \mathsf{pub}, \mathrm{VK}_j, \mathrm{VK}_i)$

P_i owns $W_i \in L(\mathsf{pub}, \mathsf{priv}_i)$ — P_j owns $W_j \in L(\mathsf{pub}, \mathsf{priv}_j)$

$(I1)$ $L_i = L(\mathsf{pub}, \mathsf{priv}_i), L'_j = L(\mathsf{pub}, \mathsf{priv}'_j)$
Randomizes W_i into V_i
$(\mathcal{C}_i, \mathcal{C}'_i) = \mathsf{EqExtCommit}(\ell_i, (\mathsf{priv}_i, \mathsf{priv}'_j, V_i); (r_i, r'_i))$
$\mathcal{C}''_i = \mathsf{EqCommit}((\mathcal{C}_i, \mathcal{C}'_i), t_i)$

flow-one $\xrightarrow{(\mathcal{C}_i, \mathcal{C}''_i)}$

$(R2)$ $L'_i = L(\mathsf{pub}, \mathsf{priv}'_i),\ L_j = L(\mathsf{pub}, \mathsf{priv}_j)$
Randomizes W_j into V_j
$\mathsf{Com}_j = \mathcal{C}_j = \mathsf{ExtCommit}(\ell_j, (\mathsf{priv}_j, \mathsf{priv}'_i, V_j); r_j)$
$\varepsilon \overset{\$}{\leftarrow},\ \mathsf{hk}_i \overset{\$}{\leftarrow} \mathsf{HashKG}(L'_i)$
$\mathsf{hp}_i = \mathsf{ProjKG}(\mathsf{hk}_i, L'_i, \mathsf{Com}_i)$
$\sigma_j = \mathsf{Sign}(\mathrm{SK}_j, (\ell_j, \mathcal{C}_j, \mathcal{C}_i, \mathcal{C}''_i, \varepsilon, \mathsf{hp}_i))$

flow-two $\xleftarrow{(\mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \sigma_j)}$

$(I3)$ Abort if
not $\mathsf{Verif}(\mathrm{VK}_j, (\ell_j, \mathcal{C}_j, \mathcal{C}_i, \mathcal{C}''_i, \varepsilon, \mathsf{hp}_i), \sigma_j)$
$z_i = r_i + \varepsilon r'_i,\ \mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$
$\mathsf{hk}_j \overset{\$}{\leftarrow} \mathsf{HashKG}(L'_j),\ \mathsf{hp}_j = \mathsf{ProjKG}(\mathsf{hk}_j, L'_j, \mathsf{Com}_j)$
$\sigma_i = \mathsf{Sign}(\mathrm{SK}_i, (\ell_i, \mathcal{C}_i, \mathcal{C}'_i, \mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \mathsf{hp}_j))$
If $W_i \notin L_i$ sets sk_i random. Otherwise,
$H_i = \mathsf{Hash}(\mathsf{hk}_j, L'_j, \ell_j, \mathsf{Com}_j)$
$H'_j = \mathsf{ProjHash}(\mathsf{hp}_i, L_i, \ell_i, \mathsf{Com}_i; z_i)$
$\mathsf{sk}_i = H_i \cdot H'_j$
Sets the session as *accepted*
and uses sk_i as a shared key

flow-three $\xrightarrow{(\mathcal{C}'_i, t_i, \mathsf{hp}_j, \sigma_i)}$

$(R4)$ Abort if
not $\mathsf{Verif}(\mathrm{VK}_i, (\ell_i, \mathcal{C}_i, \mathcal{C}'_i, \mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \mathsf{hp}_j), \sigma_i)$
or not correct opening t for $\mathcal{C}'_i$ in $\mathcal{C}''_i$
If $W_j \notin L_j$ sets sk_j random.
Otherwise, does the following:
$\mathsf{Com}_i = \mathcal{C}_i \cdot \mathcal{C}'^{\varepsilon}_i$
$H_j = \mathsf{Hash}(\mathsf{hk}_i, L'_i, \ell_i, \mathsf{Com}_i)$
$H'_i = \mathsf{ProjHash}(\mathsf{hp}_j, L_j, \ell_j, \mathsf{Com}_j; r_j)$
$\mathsf{sk}_j = H'_i \cdot H_j$
Sets the session as *accepted*
and uses sk_j as a shared key

Fig. 7. Description of the language authenticated key exchange protocol for players (P_i, ssid), with index i, message $W_i \in L_i = L(\mathsf{pub}, \mathsf{priv}_i)$ and expected language for P_j $L'_j = L(\mathsf{pub}, \mathsf{priv}'_j)$ and (P_j, ssid), with index j, message $W_j \in L_j = L(\mathsf{pub}, \mathsf{priv}_j)$ and expected language for P_i $L'_i = L(\mathsf{pub}, \mathsf{priv}'_i)$. The label is $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$. The random values used in the commitments (witnesses) are all included in (r_i, r'_i) and r_j.

D.1 Notations

The protocol is played between an initiator, denoted to as P_i, and a receiver, P_j. Each player P_k owns a public part pub_k of a language. Those two public parts pub_i and pub_j will combine to create the common public part pub of the language used in the protocol. Player P_k also owns a private part priv_k and a word $W_k \in L(\mathsf{pub}, \mathsf{priv}_k)$[1]. It rerandomizes this word W_k into a word V_k still in $L(\mathsf{pub}, \mathsf{priv}_k)$: we assume the languages used to be self-randomizable, which allows such a rerandomization.

We need three different types of commitments for this protocol:

- $\mathsf{EqCommit}$ is an equivocable commitment, such as Pedersen [Ped92], used to engage P_i on its further committed values $\mathcal{C}_i$ and $\mathcal{C}'_i$ with randomness t_i: $\mathcal{C}''_i = \mathsf{EqCommit}((\mathcal{C}_i, \mathcal{C}'_i); t_i)$;
- $\mathsf{EqExtCommit}$ is a labeled equivocable and extractable commitment, used by P_i to commit to its private values (used in the smooth projective hash function) and asking P_j to send a challenge value ε.

  It is based on a double encryption scheme (Enc_i and Enc'_i) that is partial-decryption chosen-ciphertext secure (the latter one being strongly related to the former), verifying the following properties, if we denote by $+$ and $\cdot$ two group laws adapted to the schemes:

$$\begin{aligned} \mathcal{C}_i &= \mathsf{Enc}_i(\ell_i, m_i; r_i) \\ \mathcal{C}'_i &= \mathsf{Enc}'_i(\ell_i, n_i; r'_i) \\ \mathsf{Com}_i &= \mathsf{Enc}'_i(\ell_i, m_i \cdot n_i^{\varepsilon}; r_i + \varepsilon r'_i) = \mathcal{C}_i \mathcal{C}'^{\varepsilon}_i \end{aligned}$$

  In the particular cases of (multi) Double-Cramer-Shoup or Double-Linear-Cramer-Shoup, $\mathcal{C}_i$ is a real ciphertext with the correct ξ value, to guarantee non-malleability, but $\mathcal{C}'_i$ and Com_i use the ξ value of $\mathcal{C}_i$. This is the reason why projection keys can be computed as soon as $\mathcal{C}_i$ is known.
- $\mathsf{ExtCommit}$ is a labeled extractable commitment, used by P_j to commit to its private values (used in the smooth projective hash function). It is based on a chosen-ciphertext secure encryption scheme Enc_j which can be equal to Enc_i or different: $\mathsf{Com}_j = \mathcal{C}_j = \mathsf{ExtCommit}(\ell_j, m_j; r_j) = \mathsf{Enc}_j(\ell_j, m_j; r_j)$

Again, note that the projected keys of the smooth projective hash functions depend on $\mathcal{C}_i$ and $\mathcal{C}_j$ only, and do not need Com_i, justifying it can be computed by P_j in $(R2)$, before having actually received $\mathcal{C}'_i$ and thus being able to compute Com_i.

D.2 Sketch of Proof

The proof follows that of [CHK+05] and [ACP09], but with a different approach since we want to prove that the best attack the adversary can perform is to play as an honest player would do with a chosen credential $(\mathsf{pub}_i, \mathsf{priv}_i, \mathsf{priv}'_j, W_i)$ —when trying to impersonate P_i— or $(\mathsf{pub}_j, \mathsf{priv}_j, \mathsf{priv}'_i, W_j)$ —when trying to impersonate P_j—. In order to prove Theorem 1, we need to construct, for any real-world adversary $\mathcal{A}$ (controlling some dishonest parties), an ideal-world adversary $\mathcal{S}$ (interacting with dummy parties and the split functionality $s\mathcal{F}_{\text{LAKE}}$) such that no environment $\mathcal{Z}$ can distinguish between an execution with $\mathcal{A}$ in the real world and $\mathcal{S}$ in the ideal world with non-negligible probability.

The split functionality $s\mathcal{F}_{\text{LAKE}}$ is defined in Section 5, following [BCL+05]. In particular, we assume that at the beginning of the protocol, $\mathcal{S}$ receives from it the contribution pub_i of P_i to the public language pub as answer to the Init query sent by the environment on behalf of this player. The preflow phase will determine the whole public language pub.

When initialized with security parameter k, the simulator first generates the CRS for the commitment (public parameters but also extraction and equivocation trapdoors), as well as the possibly required trapdoors to be able to generate, for any pub, a word inside or outside

[1] Since pub is unknown before the beginning of the protocol, one can imagine that P_k knows several words W_k, corresponding to different possibilities for the public part pub_ℓ its partner can choose. Once pub is set, P_k chooses a word $W_k \in L(\mathsf{pub}, \mathsf{priv}_k)$ among them or aborts the protocol if this public value does not correspond to one it had in mind.

the language $L(\mathsf{pub}, \mathsf{priv})$ when priv is known. It then initializes the real-world adversary $\mathcal{A}$, giving it these values. The simulator then starts its interaction with the environment $\mathcal{Z}$, the functionality $s\mathcal{F}_{\text{LAKE}}$ and its subroutine $\mathcal{A}$.

Since we are in the static-corruption model, the adversary can only corrupt players before the execution of the protocol. We assume players to be honest or not at the beginning, and they cannot be corrupted afterwards. However, this does not prevent the adversary from modifying flows coming from the players. Indeed, since we are in a weak authenticated setting, when a player acts dishonestly (even without being aware of it), it is either corrupted, hence the adversary knows its private values and acts on its behalf; or the adversary tries to impersonate it with chosen/guessed inputs. In both cases, we say the player is $\mathcal{A}$-controlled. Following [CHK$^+$05], we say that a flow is *oracle-generated* if it was sent by an honest player and arrives without any alteration to the player it was meant to. We say it is *non-oracle-generated* otherwise, that is if it was sent by a $\mathcal{A}$-controlled player (which means corrupted, or which flows have been modified by the adversary). The one-time signatures are aimed at avoiding changes of players during a session: if *pre-flow* is oracle-generated for P_i, then *flow-one* and *flow-three* cannot be non-oracle-generated without causing the protocol to fail because of the signature, for which the adversary does not know the signing key. Similarly, for P_j. On the other hand, if *pre-flow* is non-oracle-generated for P_i, then *flow-one* and *flow-three* cannot be oracle-generated without causing the protocol to fail, since the honest player would sign wrong flows (the flows the player sent before the adversary alters them). In both cases, the verifications of the signatures will fail at Steps $(I3)$ or $(R4)$ and P_i or P_j will abort. One can note that if there is one flow only in the protocol for one player, its signature is not required, which is the case for P_j when there is no pub to agree on at the beginning. But this is just an optimization that can be occasionally applied, as for the PAKE protocol. We do not consider it here.

To deal with both cases of $\mathcal{A}$-controlled players (either corrupted or impersonated by the adversary), we use the Split Functionality model (see Section 2). We thus add a *pre-flow* which will help us know which players are honest and which ones are $\mathcal{A}$-controlled. If one player is honest and the other one corrupted, the adversary will send the *pre-flow* on behalf of the latter, and the simulator will have to send the *pre-flow* on behalf of the former. But in the case where both players are honest at the beginning of the protocol, both *pre-flow* will have to be sent by $\mathcal{S}$ on behalf of these players and the adversary can then decide to modify one of these flows. This models the fact that the adversary can decide to split a session between P_i and P_j by answering itself to P_i, and thus trying to impersonate P_j with respect to P_i, and doing the same with P_j. Then, the Split Functionality model ensures that two independent sessions are created (with sub-session identifiers). We can thus study these sessions independently, which means that we can assume, right after the *pre-flow*, that either a player is honest if its *pre-flow* is oracle-generated, or $\mathcal{A}$-controlled if the *pre-flow* is non-oracle-generated. Since we want to show that the best possible attack for the adversary (by controlling a player) consists in playing honestly with a trial credential, we have to show that the view of the environment is unchanged if we simulate this dishonest player as an honest player with respect to ideal functionality. The simulator then has to transform its flows into queries to the Ideal Functionality $s\mathcal{F}_{\text{LAKE}}$, and namely the $\mathsf{NewSession}$-query. Still, the $\mathcal{A}$-controlled player is not honest, and can have a bad behavior when sending the real-life flows, but then either it has no strong impact, and it is similar to an honest behavior, or it will make the protocol fail: we cannot avoid the adversary to make denial of service attack, and the adversary will learn nothing.

As explained in [BCL$^+$05] and [ACGP11], where the simulator actually had access to a $\mathsf{TestPwd}$ query to the functionality, it is equivalent to grant the adversary the right to test a password (here a credential) for P_i while trying to play on behalf of P_j (i.e., use a $\mathsf{TestPwd}$ query) or to use the split functionality model and generate the $\mathsf{NewSession}$ queries corresponding to the

$\mathcal{A}$-controlled players and see how the protocol terminates, since it corresponds to a trial of one credential by the adversary (one-line dictionary attack).

The proof will thus consist in generating ideal queries (and namely the $\mathsf{NewSession}$) when receiving non-oracle-generated flows from $\mathcal{A}$-controlled players, and generating real messages for the honest players (whose $\mathsf{NewSession}$ queries will be received from the environment). This will be done in a indistinguishable way for the environment.

We assume from now on that we know in which case we are (i.e.how many players are $\mathcal{A}$-controlled), and the pub part is fixed. We then describe the simulator for each of these cases, while it has generated the *pre-flow* for the honest players by generating $(\mathrm{VK}, \mathrm{SK}) \leftarrow \mathsf{KeyGen}()$, and thus knows the signing keys. We denote by $L_i = L(\mathsf{pub}, \mathsf{priv}_i)$ the language used by P_i, and by $L'_j = L(\mathsf{pub}, \mathsf{priv}'_j)$ the language that P_i expects P_j to use. We use the same notations in the reverse direction. As explained in Section 1, recall that the languages considered depend on two possibly different relations: $L_i = L_{\mathcal{R}_i}(\mathsf{pub}, \mathsf{priv}_i)$ and $L_j = L_{\mathcal{R}_j}(\mathsf{pub}, \mathsf{priv}_j)$, but we omit them for the sake of clarity. Note that the simulator will use the NewKey query to learn whether the protocol is a success or a failure (in case a player is $\mathcal{A}$-controlled). This will enable it to check whether the LAKE should fulfill, that is, whether the two users play with compatible words and languages, i.e.. $\mathsf{priv}'_i = \mathsf{priv}_i$, $\mathsf{priv}'_j = \mathsf{priv}_j$, $W_i \in L_i$ and $W_j \in L_j$. For the most part, the interaction is implemented by the simulator $\mathcal{S}$ just following the protocol on behalf of all the honest players.

D.3 Description of the Simulators

Initialization and Simulation of *pre-flow*. This is the beginning of the simulation of the protocol, where $\mathcal{S}$ has to send the message *pre-flow* on behalf of each non-corrupted player[2].

Step ($I0$). When receiving the first $(\mathsf{Init} : \mathrm{ssid}, P_i, P_j, \mathsf{pub}_i)$ from $s\mathcal{F}_{\mathrm{LAKE}}$ as answer to the Init query sent by the environment on behalf of P_i, $\mathcal{S}$ starts simulating the new session of the protocol for party P_i, peer P_j, session identifier ssid. $\mathcal{S}$ chooses a key pair $(\mathrm{SK}_i, \mathrm{VK}_i)$ for a one-time signature scheme and generates a *pre-flow* message with the values $(\mathrm{VK}_i, \mathsf{pub}_i)$. It gives this message to $\mathcal{A}$ on behalf of (P_i, ssid).

Step ($R0$). When receiving the second $(\mathsf{Init} : \mathrm{ssid}, P_j, P_i, \mathsf{pub}_j)$ from $s\mathcal{F}_{\mathrm{LAKE}}$ as answer to the Init query sent by the environment on behalf of P_j, $\mathcal{S}$ starts simulating the new session of the protocol for party P_j, peer P_i, session identifier ssid. $\mathcal{S}$ chooses a key pair $(\mathrm{SK}_j, \mathrm{VK}_j)$ for a one-time signature scheme and generates a *pre-flow* message with the values $(\mathrm{VK}_j, \mathsf{pub}_j)$. It gives this message to $\mathcal{A}$ on behalf of (P_j, ssid).

Splitting the Players. As just said, thanks to the Split Functionality model, according to which flows were transmitted or altered by $\mathcal{A}$, we know from the *pre-flow* which player(s) is (are) honest and which player(s) is (are) $\mathcal{A}$-controlled, and the public part pub. We can consider each case independently after the initial split, during which $\mathcal{S}$ generated the signing keys of the honest players. Thanks to the signature in the last flows for each player, if the adversary tries to take control on behalf of a honest user for some part of the execution (without learning the internal states, since we exclude adaptive corruptions), the verification will fail. Then we can assume that the sent flows are the received flows.

One can note that the prior agreement on pub allows to simulate P_i before having received any information from P_j, and also without knowing whether the protocol should be a success or not. Without such an agreement, the simulator would not know which value to use for pub whereas it cannot change its mind later, since it is sent in clear. Everything else is committed: either in an equivocable way on behalf of P_i so that we can change it later when we know the real

[2] Note that $\mathcal{S}$ only has to send one of these flows if one player is corrupted.

status of the session; or in a non-equivocable way on behalf of P_j since we can check the status of the session before making this commitment. Of course, both commitments are extractable. In the whole proof, in case the extraction fails, the simulator acts as if the simulation should fail. Indeed, the language of the smooth projective hash function not only verifies the equations, but also that the ciphertext is valid, and this verification will fail.

We come back again to the case of our equivocable commitment with SPHF that is not a really extractable/binding commitment since the player can open it in a different way one would extract it, in case the second ciphertext does not encrypt $1_{\mathbb{G}}$: if extraction leads to an inconsistent tuple, there is little chance that with the random ε it becomes consistent; if extraction leads to a consistent tuple, there is little chance that with the random ε it remains consistent, and then the real-life protocol will fail, whereas the ideal-one was successful at the NewKey-time. But then, because of the positive NewKey-answer, the SendKey-query takes the key-input into consideration, that is random on the initiator side because of the SPHF on an invalid word, and thus indistinguishable from the environment point of view from a failed session: this is a denial of service, the adversary should already be aware of.

Hence, the three simulations presented below exploit the properties of our commitments and SPHF to make the view of the environment indistinguishable from a real-life attack, just using the simulator $\mathcal{S}$ that is allowed to interact with the ideal functionality on behalf of players, but in an honest way only, since the functionality is perfect and does not know bad behavior.

During all these simulations, $\mathcal{S}$ knows the equivocability trapdoor of the commitment and the decryption keys of the two encryption schemes.

Case 1: P_i is $\mathcal{A}$-controlled and P_j is honest. In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the honest player P_j, for which it has simulated the *pre-flow* and thus knows the signing key, and has to simulate the queries to the functionality as if the $\mathcal{A}$-controlled player P_i was honest.

STEP $(I1)$. This step is taken care of by the adversary, who sends its *flow-one*, from which $\mathcal{S}$ extracts $(\mathsf{priv}_i, \mathsf{priv}'_j)$ only. No need to extract W_i, but one generates a random valid $V_i \in L(\mathsf{pub}, \mathsf{priv}_i)$ (we have assumed the existence of a trapdoor in the CRS to generate such valid words). $\mathcal{S}$ sends the query $(\mathsf{NewSession} : \mathrm{ssid}', P_i, P_j, V_i, L_i = L(\mathsf{pub}, \mathsf{priv}_i), L'_j = L(\mathsf{pub}, \mathsf{priv}'_j)$, $\mathsf{initiator})$ to $\mathcal{F}_{\mathrm{LAKE}}$ on behalf of P_i.

STEP $(R2)$. The NewSession query for this player (P_j, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\mathrm{LAKE}}$ to $\mathcal{F}_{\mathrm{LAKE}}$ (transforming the session identifier from ssid to ssid$'$). $\mathcal{S}$ receives the answer $(\mathsf{NewSession} : \mathrm{ssid}, P_j, P_i, \mathsf{pub}, \mathsf{receiver})$ and makes a call NewKey to the functionality to check the success of the protocol. It actually tells whether the languages are consistent, but does not tell anything about the validity of the word submitted by the adversary for P_i. It indeed receives the answer in the name of P_i. In case of a success, $\mathcal{S}$ generates a word $V_j \in L(\mathsf{pub}, \mathsf{priv}'_j)$ and uses $\mathsf{priv}_j = \mathsf{priv}'_j$ and $\mathsf{priv}'_i = \mathsf{priv}_i$ for this receiver session (we have assumed the existence of a trapdoor in the CRS to generate such valid words) and produces a commitment $\mathcal{C}_j$ on the tuple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$. Otherwise, $\mathcal{S}$ produces a commitment $\mathcal{C}_j$ on a dummy tuple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$. It then generates a challenge value ε and the hashing keys $(\mathsf{hk}_i, \mathsf{hp}_i)$ on $\mathcal{C}_i$. It sends the *flow-two* message $(\mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \sigma_j)$ to $\mathcal{A}$ on behalf of P_j, where σ_j is the signature on all the previous information.

STEP $(I3)$. This step is taken care of by the adversary, who sends its *flow-three*.

STEP $(R4)$. Upon receiving $m = (\textit{flow-three}, \mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$, $\mathcal{S}$ makes the verification checks, and possibly aborts. In case of correct checks, $\mathcal{S}$ already knows whether the protocol should succeed, thanks to the NewKey query. If the protocol is a success, then $\mathcal{S}$ computes receiver session key honestly, and makes a SendKey to P_j. Otherwise, $\mathcal{S}$ makes a SendKey to P_j with a random key that will anyway not be used.

Case 2: P_i is honest and P_j is $\mathcal{A}$-controlled. In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the honest player P_i, for which it has simulated the *pre-flow* and thus knows the signing key, and has to simulate the queries to the functionality as if the $\mathcal{A}$-controlled player P_j was honest.

STEP ($I1$). The NewSession query for this player (P_i, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\mathrm{LAKE}}$ to $\mathcal{F}_{\mathrm{LAKE}}$ (transforming the session identifier from ssid to ssid$'$). $\mathcal{S}$ receives the answer $(\mathsf{NewSession} : \mathrm{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{initiator})$ and generates a *flow-one* message by committing to a dummy tuple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$. It gives this commitment $(\mathcal{C}_i, \mathcal{C}''_i)$ to $\mathcal{A}$ on behalf of (P_i, ssid').

STEP ($R2$). This step is taken care of by the adversary, who sends its *flow-two* $=$ (*flow-two*$, \mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \sigma_j)$, from which $\mathcal{S}$ first checks the signature, and thereafter extracts the committed triple $(\mathsf{priv}_j, \mathsf{priv}'_i, W_j)$. $\mathcal{S}$ then sends the query $(\mathsf{NewSession} : \mathrm{ssid}', P_j, P_i, W_j, L_j = L(\mathsf{pub}, \mathsf{priv}_j), L'_i = L(\mathsf{pub}, \mathsf{priv}'_i), \mathsf{receiver})$ to $\mathcal{F}_{\mathrm{LAKE}}$ on behalf of P_j.

STEP ($I3$). $\mathcal{S}$ makes a NewKey query to the functionality to know whether the protocol should succeed. It indeed receives the answer in the name of P_j. In case of a success, $\mathcal{S}$ generates a word $V_i \in L(\mathsf{pub}, \mathsf{priv}'_i)$ and uses $\mathsf{priv}_i = \mathsf{priv}'_i$ for this initiator session (we have assumed the existence of a trapdoor in the CRS to generate such valid words) and then uses the equivocability trapdoor to update $\mathcal{C}'_i$ and t in order to contain the new consistent tuple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ with respect to the challenge ε. If the protocol should be a success, then $\mathcal{S}$ computes initiator session key honestly, and makes a SendKey to P_i. Otherwise, $\mathcal{S}$ makes a SendKey to P_i with a random key that will anyway not be used. $\mathcal{S}$ sends the *flow-three* message $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to $\mathcal{A}$ on behalf of P_i, where σ_i is the signature on all the previous information.

STEP ($R4$). This step is taken care of by the adversary.

Case 3: P_i and P_j are honest. In this case, $\mathcal{S}$ has to simulate the concrete messages in the real-life from the two honest players P_i and P_j, for which it has simulated the *pre-flow* and thus knows the signing keys. But since no player is controlled by $\mathcal{A}$, the NewKey query will not provide any answer to the simulator. But thanks to the semantic security of the commitments, dummy values can be committed, no external adversary will make any difference.

STEP ($I1$). The NewSession query for this player (P_i, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\mathrm{LAKE}}$ to $\mathcal{F}_{\mathrm{LAKE}}$ (transforming the session identifier from ssid to ssid$'$). $\mathcal{S}$ receives the answer $(\mathsf{NewSession} : \mathrm{ssid}, P_i, P_j, \mathsf{pub}, \mathsf{initiator})$ and generates a *flow-one* message by committing to a dummy tuple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$. It gives this commitment $(\mathcal{C}_i, \mathcal{C}''_i)$ to $\mathcal{A}$ on behalf of (P_i, ssid').

STEP ($R2$). The NewSession query for this player (P_i, ssid') has been automatically transferred from the split functionality $s\mathcal{F}_{\mathrm{LAKE}}$ to $\mathcal{F}_{\mathrm{LAKE}}$ (transforming the session identifier from ssid to ssid$'$). $\mathcal{S}$ receives the answer $(\mathsf{NewSession} : \mathrm{ssid}, P_j, P_i, \mathsf{pub}, \mathsf{receiver})$ and generates a commitment $\mathcal{C}_j$ on a dummy tuple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$. It then generates a challenge value ε and the hashing keys $(\mathsf{hk}_i, \mathsf{hp}_i)$ on $\mathcal{C}_i$. It sends the *flow-two* message $(\mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \sigma_j)$ to $\mathcal{A}$ on behalf of P_j, where σ_j is the signature on all the previous information.

STEP ($I3$). When the session $(P_i; \mathrm{ssid}')$ receives the message $m = ($*flow-two*$, \mathcal{C}_j, \varepsilon, \mathsf{hp}_i, \sigma_j)$ from its peer session $(P_j; \mathrm{ssid}')$, the signature is necessarily correct. Then, $\mathcal{S}$ makes a SendKey to P_i with a random key that will anyway not be used, since no player is corrupted. $\mathcal{S}$ sends the *flow-three* message $(\mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ to $\mathcal{A}$ on behalf of P_i, where σ_i is the signature on all the previous information.

STEP ($R4$). When the session $(P_j; \mathrm{ssid}')$ receives the message $m = ($*flow-three*$, \mathcal{C}'_i, t, \mathsf{hp}_j, \sigma_i)$ from its peer session $(P_i; \mathrm{ssid}')$, the signature is necessarily correct. $\mathcal{S}$ makes a SendKey to P_j with a random key that will anyway not be used, since no player is corrupted.

D.4 Description of the Games

We now provide the complete proof by a sequence of games, where we replace the triple $(\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ by the notation T_i, and the triple $(\mathsf{priv}_j, \mathsf{priv}'_i, V_j)$ by the notation T_j, with component-wise operations to simplify notations. Similarly, for cleaner notations, we use non-vector notations for the ciphertexts, the random coins and the challenge ε, but all the computations are assumed to be performed component-wise, and thus implicitly use vectors.

We insist that we are considering static corruptions only, and with the split-functionality, we already know which players are corrupted and verification keys for the one-time signatures are known to the two players, and fixed: either honestly generated (honest player) or adversary-generated (corrupted players).

Game $\boldsymbol{G}_0$: This is the real game, where every flow from honest players are generated correctly by the simulator which knows the inputs sent by the environment to the players. There is no use of the ideal functionality for the moment.

Game $\boldsymbol{G}_1$: In this game, the simulator knows the decryption key for $\mathcal{C}_i$ when generating the CRS. But this game is almost the same as the previous one except the way sk_j is generated when P_i is corrupted and P_j honest. In all the other cases, the simulator does as in $\boldsymbol{G}_0$ by playing honestly (still knowing its private values). When P_i is corrupted and P_j honest, $\mathcal{S}$ does as before until $(R4)$, but then, it extracts the values committed to by the adversary in Com_i (using the decryption key for $\mathcal{C}_i$) and checks whether the private parts of the languages are consistent with the values sent to P_j by the environment. If the languages are not consistent (or decryption rejects), P_j is given a random session key sk_j.

This game is statistically indistinguishable from the former one thanks to the smoothness of the SPHF on Com_i.

Game $\boldsymbol{G}_2$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS. This game is almost the same as the previous one except that $\mathcal{S}$ extracts the values committed to by the adversary in $\mathcal{C}_i$ to check consistency of the languages, and does not wait until Com_i. If the languages are not consistent (or decryption rejects), P_j is given a random session key sk_j.

The game is indistinguishable from the previous one except if Com_i contains consistent values whereas $\mathcal{C}_i$ does not, but because of the unpredictability of ε, and the Pedersen commitment that is computationally binding under the discrete logarithm problem, the probability is bounded by $1/q$.

The distance between the two games is thus bounded by the probability to break the binding property of the Pedersen commitment.

Game $\boldsymbol{G}_3$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS, as in $\boldsymbol{G}_2$. Actually, in the above game, when P_i is corrupted and P_j honest, if extracted languages from $\mathcal{C}_i$ are not consistent, P_j does not have to compute hash values. The random coins are not needed anymore. In this game, in this particular case, $\mathcal{S}$ generates $\mathcal{C}_j$ with dummy values T'_j.

This game is computationally indistinguishable from the former one thanks to the $\mathsf{IND-CPA}$ property of the encryption scheme involved in $\mathcal{C}_j$. To prove this indistinguishability, one makes q hybrid games, where q is the number of such sessions where P_i is corrupted and P_j is honest but extracted languages from $\mathcal{C}_i$ are not consistent with inputs to P_j. More precisely, in the k-th hybrid game G_k (for $1 \leq k \leq q$), in all such sessions before the k-th one, $\mathcal{C}_j$ is generated by encrypting T'_j, in all sessions after the k-th one, $\mathcal{C}_j$ is generated by encrypting T_j, and in the k-th session, $\mathcal{C}_j$ is generated by calling the left-or-right encryption oracle on (T_j, T'_j). It is clear that the game $\boldsymbol{G}_2$ correponds to G_1 with the "left" oracle, and the game $\boldsymbol{G}_3$ corresponds to G_q with the "right" oracle. And each time, G_k with the right oracle is identical to G_{k+1} with the "left" oracle, while every game G_k is an $\mathsf{IND-CPA}$ game. It is possible to use the encryption oracle because the random coins are not needed in these sessions.

Game $\boldsymbol{G}_4$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS, as in $\boldsymbol{G}_2$. Now, when P_i is corrupted and P_j honest, if extracted languages from $\mathcal{C}_i$ are consistent, $\mathcal{S}$ knows priv_j and priv'_i (the same as the values sent by the environment). It furthermore generates a random valid word V_j, and uses it to generate the ciphertext $\mathcal{C}_j$ instead of re-randomizing the word W_j sent by the environment. $\mathcal{S}$ can compute the correct value sk_j from the random coins, and gives it to P_j.

This game is perfectly indistinguishable from the former one thanks to the self-randomizable property of the language.

Note that the value sk_j computed by $\mathcal{S}$ can be computed by the adversary if the latter indeed sent a valid word W_i in $\mathcal{C}_i$ (that is not explicitly checked in this game). Otherwise, sk_j looks random from the smoothness of the SPHF. As a consequence, on this game, sessions where P_i is corrupted and P_j is honest look ideal, while one does not need anymore the inputs from the environment sent to P_j to simulate honest players.

Game $\boldsymbol{G}_5$: We now consider the case where P_i is honest. The simulator has to simulate P_i behavior. To do so, it will know the equivocability trapdoor for the Pedersen commitment. But for other cases, the simulator still knows the decryption key for $\mathcal{C}_i$ when generating the CRS. In ($I1$), the simulator still encrypts $T_i = (\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ from the environment to produce $\mathcal{C}_i$. It chooses at random a dummy value $\mathcal{C}'_i$ and computes honestly the equivocable commitment $\mathcal{C}''_i$, knowing the random value t_i. In ($I3$), after receiving ε from P_j, it chooses random coins z_i and computes Com_i as the encryption of $T_i = (\mathsf{priv}_i, \mathsf{priv}'_j, V_i)$ with the random coins z_i. (Since this is a double encryption scheme, it uses the redundancy from $\mathcal{C}_i$: namely for DLCS, it uses ξ from $\mathcal{C}_i$). Thanks to the homomorphic property, it can compute $\mathcal{C}'_i$ as $(\mathsf{Com}_i/\mathcal{C}_i)^{1/\varepsilon}$, and equivocate $\mathcal{C}"_i$. $\mathcal{C}'_i$ should be an encryption of $1_{\mathbb{G}}$ under the random coins r'_i that are implicitly defined, but unknown.

Thanks to the properties of the different commitments recalled in Section D.1, and the perfect-hiding property of the Pedersen commitment, this is a perfect simulation. It then computes the hash values honestly, using z_i.

Game $\boldsymbol{G}_6$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i is honest, $\mathcal{S}$ generates the commitment $\mathcal{C}_i$ by choosing dummy values T'_i instead of T_i. Everything else is unchanged from $\boldsymbol{G}_5$.

This game is thus indistinguishable from the former one thanks to the $\mathsf{IND-CCA}$ property of the encryption scheme involved in $\mathcal{C}_i$. As for the proof of indistinguishability of Game $\boldsymbol{G}_3$, we do a sequence of hybrid games, where $\mathcal{C}_i$ is generated be either encrypting T_i or T'_i, or asking the left-or-right oracle on (T_i, T'_i). We replace the decryption key for $\mathcal{C}_i$ by access to the decryption oracle on $\mathcal{C}_i$. Then, one has to take care that no decryption query is asked on one of the challenge ciphertexts involved in the sequence of games. This would mean that the adversary would replay in another session a ciphertext oracle-generated in another session. Because of the label which contains the verification key oracle-generated, one can safely reject the ciphertext.

Game $\boldsymbol{G}_7$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i is honest, $\mathcal{S}$ generates the commitment $\mathcal{C}_i$ by choosing dummy values T'_i. It then computes $\mathcal{C}'_i$ by encrypting the value $(T_i/T'_i)^{1/\varepsilon}$ with randomness $z_i - r_i/\varepsilon$. This leads to the same computations of $\mathcal{C}_i$ and $\mathcal{C}'_i$ as in the former game. The rest is done as above.

This game is perfectly indistinguishable from the former one.

Game $\boldsymbol{G}_8$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i and P_j are both honest (both initiation flows where oracle-generated), if the words and languages are correct, players are both given the same random session key $\mathsf{sk}_i = \mathsf{sk}_j$. If the words and languages are not compatible, random independent session keys are given.

Since the initiation flows ($I0$ and $R0$) contained oracle-generated verification keys, unless the adversary managed to forge signatures, all the flows are oracle-generated. First, because of the pseudo-randomness of the SPHF, H_i is unpredictable, and independent of H'_j, hence sk_i looks random. Then, if the words and languages are compatible, we already has $\mathsf{sk}_j = \mathsf{sk}_i$ in the previous game. However, if they are not compatible, either H'_i is independent of H_i, or H'_j is independent of H_j, and in any case, sk_j where already independent of sk_i in the previous game.

This game is thus computationally indistinguishable from the former one, under the pseudo-randomness of the two SPHF.

Game $\boldsymbol{G}_9$: In this above game, the hash values do not have to be computed anymore when P_i and P_j are both honest. The random coins are not needed anymore.

In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. When P_i and P_j are both honest, $\mathcal{S}$ generates $\mathcal{C}'_i$ and $\mathcal{C}_j$ with dummy values T'_i and T'_j. In this game, sessions where P_i and P_j are both honest look ideal, while one does not need anymore the inputs from the environment sent to P_i and P_j to simulate honest players.

This game is computationally indistinguishable from the former one thanks to the $\mathsf{IND-CPA}$ and $\mathsf{IND-PD-CCA}$ properties of the encryption schemes involved in $\mathcal{C}_j$ and $\mathcal{C}'_i$. For the proof on indistinguishability between the two games, we make two successive sequences of hybrid games, as for the proof of indistinguishability of Game $\boldsymbol{G}_3$. One with the $\mathsf{IND-PD-CCA}$ game: a sequence of hybrid games, where $\mathcal{C}_i$ is generated by encrypting T'_i, and $\mathcal{C}'_i$ by encrypting either T_i or T'_i, but in the critical session, one asks for the left-or-right oracle $\mathsf{Encrypt}$ on (T'_i, T'_i), and the left-or-right oracle $\mathsf{Encrypt}'$ on (T_i, T'_i). The decryption key for $\mathcal{C}_i$ is replaced by an access to the decryption oracle on $\mathcal{C}_i$. As above, one has to take care that no decryption query is asked on a challenge ciphertext $\mathcal{C}'_i$, but the latter cannot be valid since it is computed from $\mathcal{C}_i$ values not controlled by the adversary. The second hybrid sequence uses $\mathsf{IND-CPA}$ games on $\mathcal{C}_j$ exactly as in the proof of indistinguishability of Game $\boldsymbol{G}_3$.

Game $\boldsymbol{G}_{10}$: In this game, the simulator still knows the decryption key for $\mathcal{C}_i$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS, but also the decryption key for $\mathcal{C}_j$. When P_i is honest and P_j corrupted, $\mathcal{S}$ extracts the values committed to by the adversary in $\mathcal{C}_j$. It checks whether they are consistent with the values sent to P_i by the environment. If the words and languages are not consistent (or decryption rejects), P_i is given a random session key sk_i.

This game is statistically indistinguishable from the former one thanks to the smoothness of the SPHF.

Game $\boldsymbol{G}_{11}$: In this game, the simulator still knows the decryption keys for $\mathcal{C}_i$ and $\mathcal{C}_j$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS.

In the above game, when P_i is honest and P_j corrupted, if extracted values from $\mathcal{C}_j$ are not consistent, P_i does not have to compute hash values. The random coins are not needed anymore. In this game, in this particular case, $\mathcal{S}$ generates $\mathcal{C}'_i$ with dummy values T'_i.

This game is computationally indistinguishable from $\boldsymbol{G}_{10}$ thanks to the $\mathsf{IND-PD-CCA}$ property of the encryption scheme involved in $\mathcal{C}'_i$. The proof uses the same sequence of hybrid games with the $\mathsf{IND-PD-CCA}$ game on $(\mathcal{C}_i, \mathcal{C}'_i)$ as in the proof of indistinguishability of Game $\boldsymbol{G}_9$.

Game $\boldsymbol{G}_{12}$: In this game, the simulator still knows the decryption keys for $\mathcal{C}_i$ and $\mathcal{C}_j$ and the equivocability trapdoor for the Pedersen commitment when generating the CRS. Now, when P_i is honest and P_j corrupted, if extracted values from $\mathcal{C}_j$ are consistent, $\mathcal{S}$ knows priv_i and priv'_j

DLin	$\mathbb{G}$	$\mathbb{Z}_p$	Exp.	CSCom	$\mathbb{G}$	$\mathbb{Z}_p$	Exp.
LCSCom	$5n$	0	$7n+2$	CSCom	$4n$	0	$4n+1$
DLCSCom	$10n+1$	2	$18n+6$	DCSCom	$8n+1$	2	$12n+5$
Equality	2	0	14	Equality	1	0	10
LPPE	$2n+1$	0	$10n+11$	LPPE	$n+1$	0	$7n+9$

Table 1. Computational and Communication Costs

(the same as the values sent by the environment). It furthermore generates a random valid word V_i, and uses it to generate the ciphertext $\mathcal{C}'_i$ instead of re-randomizing the word W_j sent by the environment. $\mathcal{S}$ can compute the correct value sk_i from the random coins, and gives it to P_i. In this game, sessions where P_i is honest and P_j is corrupted look ideal, while one does not need anymore the inputs from the environment sent to P_i to simulate honest players.

This game is perfectly indistinguishable from the former one thanks to the self-randomizable property of the language.

Game G_{13}: In this game, $\mathcal{S}$ now uses the ideal functionality: NewSession-queries for honest players are automatically forwarded to the ideal functionality, for corrupted players, they are done by $\mathcal{S}$ using the values extracted from $\mathcal{C}_i$ or $\mathcal{C}_j$. In order to check consistency of the words and languages, $\mathcal{S}$ asks for a NewKey. When one player is corrupted, it learns the outcome: success or failure. It can continue the simulation in an appropriate way.

E Complexity

In the Table 1, we give the number of elements to be sent (group elements or scalars) and the number of exponentiations to compute for each operation (commitment and SPHF), where we consider the Equality Test, and the Linear Pairing Product Equations. One has to commit all the private inputs, and then the cost for relations is just the additional overhead due to the projection keys and hashing computations once the elements are already committed: an LCSCom commitment is 5 group elements, and a DLCSCom$'$ is twice more, plus the Pedersen commitment (one group element), the challenge ε (a scalar) and the opening t (a scalar). Note that a simple Linear commitment is just 3 group elements.

If the global language is a conjunction of several languages, one should simply add all the costs, and consider the product of all the sub-hashes as the final hash value from the SPHF.

PAKE. Two users want to prove to each other they possess the same password. In this case $W_i = \mathsf{priv}'_j = \mathsf{priv}_i = \mathsf{priv}_j = \mathsf{priv}'_i = W_j$. So P_i will commit to his password, and thus a unique DLCSCom commitment for W_i, priv_i and priv'_i. P_j can use a simple Linear commitment. They then send projection keys for equality tests: 13 group elements and 2 scalars for Com_i and 5 group elements for Com_j, plus VK_i and σ_i. This leads to 18 group elements and two scalars our PAKE scheme. The DDH-based variant would use 11 group elements and 2 scalars only in total, which is far more efficient than existing solutions, and namely [ACP09] that uses a bit-per-bit commitment to provide equivocability.

Verifier-based PAKE. As explained earlier, we do a PAKE with the common password $(g^{\mathsf{pw}}, h^{\mathsf{pw}})$, where h has been chosen by the server: the commitment Com_i needs 21 group elements plus 2 scalars, and 4 additional group elements to check it; the commitment Com_j needs 6 group elements, and 4 additional elements to check it. Because of the ephemeral h, one has to send in total 35 group elements and 2 scalars, plus the one-time signatures. The DDH-based variant would use 25 group elements and 2 scalars only in total.

Secret Handshake. The users want to check their partner possesses a valid signature on their public identity or pseudonym (in pub) under some valid but private verification key (affiliation-hiding). More precisely, P_i wants to prove he possesses a valid signature σ on the public message m (his identity or a pseudonym) under a private verification key vk: we thus have m in the pub part, $\mathsf{priv}_i = \mathsf{vk}$ and $W = \sigma$. This is the same for P_j. Using Waters signature, $\sigma = (\sigma_1, \sigma_2)$, where σ_1 only has to be encrypted, because σ_2 does not contain any information, it can thus be sent in clear. In addition, as noticed from the security proof, σ_2 does not need to be encrypted in an $\mathsf{IND-PD-CCA}$ manner, but with a simple $\mathsf{IND-CPA}$ encryption scheme in the third round. To achieve unlinkability, one can rerandomize this signature σ to make the σ_2 values different and independent each time.

As a consequence, the committed values are: vk that can be any group element, since with the master secret key s such that $h = g^s$ for the global parameters of the Waters signature (see the Appendix A.3) one can derive the signing key associated to any verification key, and thus generate a valid word in the language; and σ_1 in $\mathsf{IND-CPA}$ only. One additionally sends σ_2 in clear, and so 14 group elements plus 2 scalars for Com_i, and 7 group elements for Com_j. The languages to be verified are $\mathsf{priv}_i = \mathsf{priv}'_i$, on the committed $\mathsf{priv}_i = \mathsf{vk}_i$ with the expected $\mathsf{priv}'_i = \mathsf{vk}'_i$, and the Linear Pairing Product Equation for the committed signature σ_i, but under the expected vk'_i: 5 group elements for the projection keys in both directions: 31 group elements plus 2 scalars are sent in total.

Appendix B

New Techniques for SPHFs and Efficient One-Round PAKE Protocols [BBC^{+}13b]

This is the Full Version of the part on Smooth Projective Hash Functions of the Extended Abstract "New Techniques for SPHFs and Efficient One-Round PAKE Protocols" that appears in Advances in Cryptology – Proceedings of CRYPTO 2013 (18–22 august 2013, Santa Barbara, California, USA), R.Canetti and J. A. Garay Eds. Springer-Verlag, LNCS 8042, pages 449–475.

Authors

Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, Damien Vergnaud

Abstract

Password-Authenticated Key Exchange (PAKE) has received deep attention in the last few years, with a recent improvement by Katz and Vaikuntanathan, and their one-round protocols: the two players just have to send simultaneous flows to each other, that depend on their own passwords only, to agree on a shared high entropy secret key. To this aim, they followed the Gennaro and Lindell's approach, with a new kind of *Smooth-Projective Hash Functions* (SPHFs). They came up with the first concrete one-round PAKE, secure in the Bellare, Pointcheval, and Rogaway's model, but at the cost of a simulation-sound NIZK, which makes the overall construction not really efficient.

This paper follows their path with a new efficient instantiation of SPHF on Cramer-Shoup ciphertexts. It then leads to the design of the most efficient PAKE known so far: a one-round PAKE with two simultaneous flows consisting of 6 group elements each only, in any DDH-group without any pairing. We thereafter show a generic construction for SPHFs, in order to check the validity of complex relations on encrypted values. This allows to extend this work on PAKE to the more general family of protocols, termed *Langage-Authenticated Key Exchange* (LAKE) by Ben Hamouda, Blazy, Chevalier, Pointcheval, and Vergnaud, but also to blind signatures. We indeed provide the most efficient blind Waters' signature known so far.

1 Introduction

Authenticated Key Exchange protocols are quite important primitives for practical applications, since they enable two parties to generate a shared high entropy secret key, to be later used with symmetric primitives in order to protect communications, while interacting over an insecure network under the control of an adversary. Various authentication means have been proposed, and the most practical one is definitely a shared low entropy secret, or a password they can agree on over the phone, hence PAKE, for *Password-Authenticated Key Exchange*. The most famous instantiation has been proposed by Bellovin and Merritt [BM92], EKE for Encrypted Key Exchange, which simply consists of a Diffie-Hellman key exchange [DH76], where the flows are symmetrically encrypted under the shared password. Overall, the equivalent of 2 group elements have to be sent.

A first formal security model was proposed by Bellare, Pointcheval and Rogaway [BPR00] (the BPR model), to deal with off-line dictionary attacks. It essentially says that the best attack should be the on-line exhaustive search, consisting in trying all the passwords by successive executions of the protocol with the server. Several variants of EKE with BPR-security proofs have been proposed in the ideal-cipher model or the random-oracle model [Poi12]. Katz, Ostrovsky and Yung [KOY01] proposed the first practical scheme (KOY), provably secure in the standard model under the DDH assumption. This is a 3-flow protocol, with the client sending 5 group elements plus a verification key and a signature, for a one-time signature scheme, and the server sending 5 group elements. It has been generalized by Gennaro and Lindell [GL03] (GL), making use of smooth projective hash functions.

Smooth Projective Hash Functions (SPHFs) were introduced by Cramer and Shoup [CS02] in order to achieve IND-CCA security from IND-CPA encryption schemes, which led to the first efficient IND-CCA encryption scheme provably secure in the standard model under the DDH assumption [CS98]. They can be seen as a kind of implicit designated-verifier proofs of membership [ACP09, BPV12]. Basically, SPHFs are families of pairs of functions $(\mathsf{Hash}, \mathsf{ProjHash})$ defined on a language L. These functions are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where hk, the hashing key, can be seen as the private key and hp, the projection key, as the public key. On a word $W \in \mathrm{L}$, both functions should lead to the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ with the hashing key and $\mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, W, w)$ with the projection key only but also a witness w that $W \in \mathrm{L}$. Of course, if $W \notin \mathrm{L}$, such a witness does not exist, and the smoothness property states that $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ is independent of hp. As a consequence, even knowing hp, one cannot guess $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$.

One-Round PAKE in the BPR Model. Gennaro and Lindell [GL03] (GL) extended the initial definition of smooth projective hash functions for an application to PAKE. Their approach has thereafter been adapted to the *Universal Composability* (UC) framework by Canetti *et al.* [CHK+05], but for static corruptions only. It has been improved by Abdalla, Chevalier and Pointcheval [ACP09] to resist to adaptive adversaries. But the 3-flow KOY protocol remains the most efficient protocol BPR-secure under the DDH assumption.

More recently, the ultimate step for PAKE has been achieved by Katz and Vaikuntanathan [KV11] (KV), who proposed a *practical* one-round PAKE, where the two players just have to send simultaneous flows to each other, that depend on their own passwords only. More precisely, each flow just consists of an IND-CCA ciphertext of the password and an SPHF projection key for the correctness of the partner's ciphertext (the word is the ciphertext and the witness consists of the random coins of the encryption). The shared secret key is eventually the product of the two hash values, as in the KOY and GL protocols. Because of the simultaneous flows, one flow cannot explicitly depend on the partner's flow, which makes impossible the use of the Gennaro and Lindell SPHF (later named GL-SPHF), in which the projection key depends on the word (the ciphertext here). On the other hand, the adversary can wait for the player to send his flow first, and then adapt its message, which requires stronger security notions than the initial Cramer and Shoup SPHF (later named CS-SPHF), in which the smoothness does not hold anymore if the word is generated after having seen the projection key. This led Katz and Vaikuntanathan to provide a new definition for SPHF (later named KV-SPHF), where the projection key depends on the hashing key only, and the smoothness holds even if the word is chosen after having seen the projection key. Variations between CS-SPHF, GL-SPHF and KV-SPHF are in the way one computes the projection key hp from the hashing key hk and the word W, but also in the smoothness property, according to the freedom the adversary has to choose W, when trying to distinguish the hash value from a random value. As a side note, while CS-SPHF is close to the initial definition, useful for converting an IND-CPA encryption scheme to IND-CCA, GL-SPHFs and KV-SPHFs did prove quite useful too: we will use KV-SPHFs for our one-round PAKE protocols and a GL-SPHF for the blind signature scheme.

As just explained, the strongest definition of SPHF, which gives a lot of freedom to the adversary, is the recent KV-SPHF. However, previous SPHFs known on Cramer-Shoup ciphertexts were GL-SPHFs only. For their one-round PAKE, Katz and Vaikuntanathan did not manage to construct such a KV-SPHF for an efficient IND-CCA encryption scheme. They then suggested to use the Naor and Yung approach [NY90], with an ElGamal-like encryption scheme and a *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proof [Sah99]. Such an SS-NIZK proof is quite costly in general. They suggested to use Groth-Sahai [GS08] proofs in bilinear groups and the linear encryption [BBS04] which leads to a PAKE secure under the DLin assumption with a ciphertext consisting of 66 group elements and a projection key consisting of 4 group elements. As a consequence, the two players have to send 70 group elements each, which is far more costly than the KOY protocol, but it is one-round only.

More recent results on SS-NIZK proofs or IND-CCA encryption schemes, in the discrete logarithm setting, improved on that: Libert and Yung [LY12] proposed a more efficient SS-NIZK proof of plaintext equality in the Naor-Yung-type cryptosystem with ElGamal-like encryption. The proof can be reduced from 60 to 22 group elements and the communication complexity of the resulting PAKE is decreased to 32 group elements per user. Jutla and Roy [JR12] proposed relatively-sound NIZK proofs as an efficient alternative to SS-NIZK proofs to build new publicly-verifiable IND-CCA encryption schemes. They can then decrease the PAKE communication complexity to 20 group elements per user. In any case, one can remark that all one-round PAKE schemes require pairing computations.

Language-Authenticated Key Exchange. A generalization of AKE protocols has been recently proposed, so-called *Language-Authenticated Key Exchange* (LAKE) [BBC+13]: it allows two users, Alice and Bob, each owning a word in a specific language, to agree on a shared high entropy secret if each user knows a word in the language the other thinks about. More precisely, they first both agree on public parameters pub, Bob will think about *priv* for his expected Alice's value of priv, Alice will do the same with *priv′* for Bob's private value priv′; eventually, if *priv* = priv and *priv′* = priv′, and if they both know words in the appropriate languages, then the key agreement will succeed. In case of failure, no information should leak to the players about the reason of failure, except that the inputs did not satisfy the relations, or the languages were not consistent. Eavesdroppers do not even learn the outcome.

This formalism encompasses PAKE, and their first construction follows the GL approach for PAKE: each player commits to the private values (his own value priv, and his expected partner's value *priv′*) as well as his own word, and projection keys are sent to compute random values that will be the same if and only if everything is consistent. To achieve one-round LAKE, one also needs KV-SPHF on ciphertexts for plaintext-equality tests (equality of the private values and expected private values) and for language-membership.

Achievements. Our main contribution is the description of an instantiation of KV-SPHF on Cramer-Shoup ciphertexts, and thus the first KV-SPHF on an efficient IND-CCA encryption scheme. We thereafter use it within the above KV framework for one-round PAKE [KV11], in the

BPR security model. Our scheme just consists of 6 group elements in each direction under the DDH assumption (4 for the ciphertext, and 2 for the projection key). This has to be compared with the 20 group elements, or more, in the best constructions discussed above, which all need pairing-friendly groups and pairing computations, or with the KOY protocol that has a similar complexity but with three sequential flows.

We also present the first GL-SPHFs/KV-SPHFs able to handle multi-exponentiation equations without requiring pairings. Those SPHFs are thus quite efficient. They lead to two applications. First, our new KV-SPHFs enable several efficient instantiations of one-round *Language-Authenticated Key-Exchange* (LAKE) protocols [BBC+13]. Our above one-round PAKE scheme is actually a particular case of a more general one-round LAKE scheme, for which we provide a BPR-like security model and a security proof. Our general constructions also cover Credential-Authenticated Key Exchange [CCGS10]. Second, thanks to a new GL-SPHF, we improve on the *blind signature* scheme presented in [BPV12], from $5\ell + 6$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$ to $3\ell + 7$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$, for an ℓ-bit message to be blindly signed with a Waters signature [Wat05]. Our protocol is round-optimal, since it consists of two flows, and leads to a classical short Waters signature.

As a side contribution, we introduce a new generic framework to construct SPHFs aiming at making easier the construction and the proof of SPHFs on complex languages. Using this framework, we were able to construct SPHFs for any language handled by the Groth-Sahai NIZK proofs, and so for any $\mathcal{NP}$-language. The main idea of this framework consists in considering languages as subspaces of some large vector space over cyclic groups and is already present in Section 7.4.1 of the full version of [CS02]. Our generic framework is slightly more general as it allows us to consider bilinear and multilinear groups, and not just cyclic groups.

Outline of the Paper. In Section 2, we first revisit the different definitions for SPHFs proposed in [CS02, GL03, KV11], denoted respectively CS-SPHFs, GL-SPHFs and KV-SPHFs. While CS-SPHF was the initial definition useful for converting an IND-CPA encryption scheme to IND-CCA, GL-SPHFs and KV-SPHFs did prove quite useful too: we will use a KV-SPHF for our PAKE/LAKE application and a GL-SPHF for the blind signature. In Section 2.4, we introduce our main contribution, the construction of a KV-SPHF on Cramer-Shoup ciphertexts. This KV-SPHF leads to the construction of our efficient one-round PAKE in Section 2.5. In Section 3, we present a simplified version of our generic framework (fully described in Appendix D). We then show our efficient SPHFs on multi-exponentiation equations and on bit encryption, without pairings, in Section 4. Finally, in Section 5, we introduce our two other constructions based on these SPHFs: our one-round LAKE and our blind signature scheme.

2 New SPHF on Cramer-Shoup Ciphertexts and PAKE

In this section, we first recall the definitions of SPHFs and present our classification based on the dependence between words and keys. According to this classification, there are three types of SPHFs: the (almost) initial Cramer and Shoup [CS02] type (CS-SPHF) introduced for enhancing an IND-CPA encryption scheme to IND-CCA, the Gennaro and Lindell [GL03] type (GL-SPHF) introduced for PAKE, and the Katz and Vaikuntanathan [KV11] type (KV-SPHF) introduced for one-round PAKE.

Then, after a quick review on the Cramer-Shoup encryption scheme, we introduce our new KV-SPHF on Cramer-Shoup ciphertexts which immediately leads to a quite efficient instantiation of the Katz and Vaikuntanathan one-round PAKE [KV11], secure in the BPR model.

2.1 General Definition of SPHFs

Let us consider a language $\mathrm{L} \subseteq \mathcal{S}et$, and some global parameters for the SPHF, assumed to be in the common random string (CRS). The SPHF system for the language L is defined by four algorithms:

- $\mathsf{HashKG}(\mathrm{L})$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C)$ derives the projection key hp, possibly depending on the word C;
- $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)$ outputs the hash value of the word C from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, C, w)$ outputs the hash value of the word C from the projection key hp, and the witness w that $C \in \mathrm{L}$.

The *correctness* of the SPHF assures that if $C \in \mathrm{L}$ with w a witness of this membership, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, C, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $C \notin \mathrm{L}$, the hash value is *statistically* indistinguishable from a random element, even knowing hp. For that, we use the classical notion of statistical distance recalled in Appendix A.2.

2.2 Smoothness Adaptivity and Key Word-Dependence

This paper will exploit the very strong notion KV-SPHF. Informally, while the GL-SPHF definition allows the projection key hp to depend on the word C, the KV-SPHF definition prevents the projection key hp from depending on C, as in the original CS-SPHF definition. In addition, the smoothness should hold even if C is chosen as an arbitrary function of hp. This models the fact the adversary can see hp before deciding which word C it is interested in. More formal definitions follow, where we denote Π the range of the hash function.

CS-SPHF. This is almost[1] the initial definition of SPHF, where the projection key hp does not depend on the word C (word-independent key), but the word C cannot be chosen after having seen hp for breaking the smoothness (non-adaptive smoothness). More formally, a CS-SPHF is ε-smooth if ProjKG does not use its input C and if, for any $C \in \mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \bot);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)\}$$
$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \bot);\ H \overset{\$}{\leftarrow} \Pi\}.$$

GL-SPHF. This is a relaxation, where the projection key hp can depend on the word C (word-dependent key). More formally, a GL-SPHF is ε-smooth if, for any $C \in \mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)\}$$
$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C);\ H \overset{\$}{\leftarrow} \Pi\}.$$

KV-SPHF. This is the strongest SPHF, in which the projection key hp does not depend on the word C (word-independent key) and the smoothness holds even if C depends on hp (adaptive smoothness). More formally, a KV-SPHF is ε-smooth if ProjKG does not use its input C and, for any function f onto $\mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \bot);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, f(\mathsf{hp}))\}$$
$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \bot);\ H \overset{\$}{\leftarrow} \Pi\}.$$

[1] In the initial definition, the smoothness was defined for a word C randomly chosen from $\mathcal{S}et \backslash \mathrm{L}$, and not necessarily for any such word.

Remark 1. One can see that a perfectly smooth (*i.e.*, 0-smooth) $\mathsf{CS\text{-}SPHF}$ is also a perfectly smooth $\mathsf{KV\text{-}SPHF}$, since each value H has exactly the same probability to appear, and so adaptively choosing C does not increase the above statistical distance. However, as soon as a weak word C can bias the distribution, f can exploit it.

2.3 SPHFs on Languages of Ciphertexts

We could cover languages as general as those proposed in [BBC+13], but for the sake of clarity, and since the main applications need some particular cases only, we focus on SPHFs for languages of ciphertexts, whose corresponding plaintexts verify some relations. We denote these languages $\textsc{LofC}_{\mathsf{full\text{-}aux}}$.

The parameter $\mathsf{full\text{-}aux}$ will parse in two parts $(\mathsf{crs}, \mathsf{aux})$: the public part crs, known in advance, and the private part aux, possibly chosen later. More concretely, crs represents the public values: it will define the encryption scheme (and will thus contain the global parameters and the public key of the encryption scheme) with the global format of both the tuple to be encrypted and the relations it should satisfy, and possibly additional public coefficients; while aux represents the private values: it will specify the relations, with more coefficients or constants that will remain private, and thus implicitly known by the sender and the receiver (as the expected password, for example, in PAKE protocols).

To keep aux secret, hp should not leak any information about it. We will thus restrict HashKG and ProjKG not to use the parameter aux, but just crs. This is a stronger restriction than required for our purpose, since one can use aux without leaking any information about it. But we already have quite efficient instantiations, and it makes everything much simpler to present.

2.4 SPHFs on Cramer-Shoup Ciphertexts

Labeled Cramer-Shoup Encryption Scheme (CS). The CS labeled encryption scheme is recalled in Appendix A.3. We briefly review it here. We combine all the public information in the encryption key. We thus have a group $\mathbb{G}$ of prime order p, with two independent generators $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, a hash function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$ from a collision-resistant hash function family onto $\mathbb{Z}_p^*$, and a reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$. From 5 scalars $(x_1, x_2, y_1, y_2, z) \xleftarrow{\$} {\mathbb{Z}_p}^5$, one also sets $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. The encryption key is $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$, while the decryption key is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z)$. For a message $m \in \{0,1\}^n$, with $M = \mathcal{G}(m) \in \mathbb{G}$, the labeled Cramer-Shoup ciphertext is:

$$C \stackrel{\mathsf{def}}{=} \mathsf{CS}(\ell, \mathsf{ek}, M; r) \stackrel{\mathsf{def}}{=} (\mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r),$$

with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e) \in \mathbb{Z}_p^*$. If one wants to encrypt a vector of group elements $(M_1, \dots, M_n)$, all at once in a non-malleable way, one computes all the individual ciphertexts with a common $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \dots, \mathbf{u}_n, e_1, \dots, e_n)$ for $v_1, \dots, v_n$. Hence, everything done on tuples of ciphertexts will work on ciphertexts of vectors. In addition, the Cramer-Shoup labeled encryption scheme on vectors is $\mathsf{IND\text{-}CCA}$ under the DDH assumption.

The (known) GL-SPHF for CS. Gennaro and Lindell [GL03] proposed an SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^4$. The associated projection key, on a ciphertext $C = (\mathbf{u} = (u_1, u_2) = (g_1^r, g_2^r), e = \mathcal{G}(m) \cdot h^r, v = (cd^\xi)^r)$, is $\mathsf{hp} = g_1^\eta g_2^\theta h^\mu (cd^\xi)^\nu \in \mathbb{G}$. Then, one can compute the hash value in two different ways, for the language $\textsc{LofC}_{\mathsf{ek},m}$ of the valid ciphertexts of $M = \mathcal{G}(m)$, where $\mathsf{crs} = \mathsf{ek}$ is public but $\mathsf{aux} = m$ is kept secret:

$$\begin{aligned} H \stackrel{\mathsf{def}}{=} \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\mathsf{def}}{=} u_1^\eta u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu \\ &= \mathsf{hp}^r \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) \stackrel{\mathsf{def}}{=} H'. \end{aligned}$$

- Players U and U' both use $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$;
- U, with password pw, chooses $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$,
  computes $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu})$, sets $\ell = (U, U', \mathsf{hp})$,
  and generates $C = (\mathbf{u} = (g_1^r, g_2^r), e = \mathcal{G}(\mathsf{pw}) \cdot h^r, v = (cd^{\xi})^r)$ with r a random scalar in $\mathbb{Z}_p$ and $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
  U sends $\mathsf{hp} \in \mathbb{G}^2$ and $C \in \mathbb{G}^4$ to U';
- Upon receiving $\mathsf{hp}' = (\mathsf{hp}'_1, \mathsf{hp}'_2) \in \mathbb{G}^2$ and $C' = (\mathbf{u}' = (u'_1, u'_2), e', v') \in \mathbb{G}^4$ from U', U sets $\ell' = (U', U, \mathsf{hp}')$ and $\xi' = \mathfrak{H}_K(\ell', \mathbf{u}', e')$ and computes
  $$\mathsf{sk}_U = {u'_1}^{(\eta_1 + \xi' \eta_2)} {u'_2}^{\theta} (e'/\mathcal{G}(\mathsf{pw}))^{\mu} {v'}^{\nu} \cdot (\mathsf{hp}'_1 {\mathsf{hp}'_2}^{\xi})^r.$$

Fig. 1. One-Round PAKE based on DDH

A (new) KV-SPHF for CS. We give here the description of the first known KV-SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$; the associated projection key is the pair $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu}) \in \mathbb{G}^2$. Then one can compute the hash value in two different ways, for the language $\mathrm{LOFC}_{\mathsf{ek},m}$ of the valid ciphertexts of $M = \mathcal{G}(m)$ under ek:

$$\begin{aligned} H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi \eta_2)} u_2^{\theta} (e/\mathcal{G}(m))^{\mu} v^{\nu} \\ &= (\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = H'. \end{aligned}$$

Theorem 2. *The above* SPHF *is a perfectly smooth (*i.e., *0-smooth)* KV-SPHF.

The proof can be found in Section D.3 as an illustration of our new framework.

2.5 An Efficient One-Round PAKE

Review of the Katz and Vaikuntanathan's PAKE. As explained earlier, Katz and Vaikuntanathan recently proposed a one-round PAKE in [KV11]. Their general framework follows Gennaro and Lindell [GL03] approach: each player sends an encryption of the password, and then uses an SPHF on the partner's ciphertext to check whether it actually contains the same password. The two hash values are multiplied to produce the session key. If the encrypted passwords are the same, the different ways to compute the hash values (Hash and ProjHash) give the same results. If the passwords differ, the smoothness makes the values computed by each player independent. To this aim, the authors need an SPHF on a labeled IND-CCA encryption scheme. To allow a SPHF-based PAKE scheme to be one-round, the ciphertext and the SPHF projection key for verifying the correctness of the partner's ciphertext should be sent together, before having seen the partner's ciphertext: the projection key should be independent of the ciphertext. In addition, the adversary can wait until it receives the partner's projection key before generating the ciphertext, and thus a stronger smoothness is required. This is exactly why we need a KV-SPHF in this one-round PAKE framework.

Our Construction. Our KV-SPHF on Cramer-Shoup ciphertexts can be used in the Katz and Vaikuntanathan framework for PAKE [KV11]. It leads to the most efficient PAKE known so far, and it is *one-round.* Each user indeed only sends 6 elements of $\mathbb{G}$ (see Figure 1), instead of 70 elements of $\mathbb{G}$ for the Katz and Vaikuntanathan's instantiation using a Groth-Sahai SS-NIZK [GS08], or 20 group elements for the Jutla and Roy's [JR12] improvement using a relatively-sound NIZK.

The formal security result follows from the Theorem 4 in Section 5.1. We want to insist that our construction does not need pairing-friendly groups, and the plain DDH assumption is enough, whereas the recent constructions made heavy use of pairing-based proofs *à la* Groth-Sahai. Under the DLin assumption (which is a weaker assumption in any group), still without requiring pairing-friendly groups, our construction would make each user to send 9 group elements only.

3 Generic Framework for SPHFs

3.1 Introduction

In Appendix D, we propose a formal framework for SPHFs using a new notion of graded rings, derived from [GGH12]. It enables to deal with cyclic groups, bilinear groups (with symmetric or asymmetric pairings), or even groups with multi-linear maps. In particular, it helps to construct concrete SPHFs for quadratic pairing equations over ciphertexts, which enable to construct efficient LAKE [BBC+13] for any language handled by the Groth-Sahai NIZK proofs, and so for any $\mathcal{NP}$-language (see Section 5.1).

However, we focus here on cyclic groups, with the basic intuition only, and provide some illustrations. While we keep the usual multiplicative notation for the cyclic group $\mathbb{G}$, we use an extended notation: $r \odot u = u \odot r = u^r$, for $r \in \mathbb{Z}_p$ and $u \in \mathbb{G}$, and $u \oplus v = u \cdot v$, for $u, v \in \mathbb{G}$. Basically, $\oplus$ and $\odot$ correspond to the addition and the multiplication in the exponents, that are thus both commutative. We then extend this notation in a natural way when working on vectors and matrices.

Our goal is to deal with languages of ciphertexts $\textsc{LofC}_{\mathsf{full\text{-}aux}}$: we assume that crs is fixed and we write $\mathrm{L}_{\mathsf{aux}} = \textsc{LofC}_{\mathsf{full\text{-}aux}} \subseteq \mathcal{S}et$ where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$.

3.2 Language Representation

For a language $\mathrm{L}_{\mathsf{aux}}$, we assume there exist two positive integers k and n, a function $\Gamma : \mathcal{S}et \mapsto \mathbb{G}^{k\times n}$, and a family of functions $\Theta_{\mathsf{aux}} : \mathcal{S}et \mapsto \mathbb{G}^{1\times n}$, such that for any word $C \in \mathcal{S}et$, $(C \in \mathrm{L}_{\mathsf{aux}}) \iff (\exists \boldsymbol{\lambda} \in \mathbb{Z}_p^{1\times k}$ such that $\Theta_{\mathsf{aux}}(C) = \boldsymbol{\lambda} \odot \Gamma(C))$. In other words, we assume that $C \in \mathrm{L}_{\mathsf{aux}}$, if and only if, $\Theta_{\mathsf{aux}}(C)$ is a linear combination of (the exponents in) the rows of some matrix $\Gamma(C)$. For a KV-SPHF, Γ is supposed to be a constant function (independent of the word C). Otherwise, one gets a GL-SPHF.

We furthermore require that a user, who knows a witness w of the membership $C \in \mathrm{L}_{\mathsf{aux}}$, can efficiently compute the above *linear* combination $\boldsymbol{\lambda}$. This may seem a quite strong requirement but this is actually verified by very expressive languages over ciphertexts such as ElGamal, Cramer-Shoup and variants.

We briefly illustrate it on our KV-SPHF on CS: $C = (u_1 = g_1^r, u_2 = g_2^r, e = M \cdot h^r, v = (cd^\xi)^r)$, with $k = 2$, $\mathsf{aux} = M$ and $n = 5$:

$$\Gamma = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \qquad \boldsymbol{\lambda} = (r, r\xi) \qquad \begin{aligned} \boldsymbol{\lambda} \odot \Gamma &= (g_1^r, g_1^{r\xi}, g_2^r, h^r, (cd^\xi)^r) \\ \Theta_M(C) &= (u_1, u_1^\xi, u_2, e/M, v). \end{aligned}$$

Essentially, one tries to make the first columns of $\Gamma(C)$ and the first components of $\Theta_{\mathsf{aux}}(C)$ to completely determine $\boldsymbol{\lambda}$. In our illustration, the first two columns with $u_1 = g_1^r$ and $u_1^\xi = g_1^{r\xi}$ really imply $\boldsymbol{\lambda} = (r, r\xi)$, and the three last columns help to check the language membership: we want $u_2 = g_2^r$, $e/M = h^r$, and $v = (cd^\xi)^r$, with the same r as for u_1.

3.3 Smooth Projective Hash Function

With the above notations, the hashing key is a vector $\mathsf{hk} = \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\intercal \stackrel{\$}{\leftarrow} \mathbb{Z}_p^n$, while the projection key is, for a word C, $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \odot \boldsymbol{\alpha} \in \mathbb{G}^k$ (if Γ depends on C, this leads to a GL-SPHF, otherwise, one gets a KV-SPHF). Then, the hash value is:

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \stackrel{\text{def}}{=} \Theta_{\mathsf{aux}}(C) \odot \boldsymbol{\alpha} = \boldsymbol{\lambda} \odot \boldsymbol{\gamma}(C) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w).$$

Our above Γ, $\boldsymbol{\lambda}$, and Θ_M immediately lead to our KV-SPHF on CS from the Section 2.4: with $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^5$, the product with Γ leads to: $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu}) \in \mathbb{G}^2$, and

$$\begin{aligned} H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\text{def}}{=} u_1^{(\eta_1+\xi\eta_2)} u_2^{\theta} (e/\mathcal{G}(m))^{\mu} v^{\nu} \\ &= (\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = H'. \end{aligned}$$

The generic framework detailed in Appendix D also contains a security analysis that proves the above generic SPHF is perfectly smooth: Intuitively, for a word $C \notin \mathrm{L}_{\mathsf{aux}}$ and a projection key $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \odot \boldsymbol{\alpha}$, the vector $\Theta_{\mathsf{aux}}(C)$ is not in the linear span of $\Gamma(C)$, and thus $H = \Theta_{\mathsf{aux}}(C) \odot \boldsymbol{\alpha}$ is independent from $\Gamma(C) \odot \boldsymbol{\alpha} = \mathsf{hp}$.

4 Concrete Constructions of SPHFs

In this section, we illustrate more our generic framework, by constructing more evolved SPHFs without pairings. More complex constructions of SPHFs, namely for any language handled by the Groth-Sahai NIZK proofs, are detailed in Appendix D.

4.1 KV-SPHF for Linear Multi-Exponentiation Equations

We present several instantiations of KV-SPHFs, in order to illustrate our framework, but also to show that our one-round PAKE protocol from Section 2.5 can be extended to one-round LAKE [BBC+13]. In PAKE/LAKE, we use SPHFs to prove that the plaintexts associated with some ElGamal-like ciphertexts verify some relations. The communication complexity of these protocols depends on the ciphertexts size and of the projection keys size. We first focus on ElGamal ciphertexts, and then explain how to handle Cramer-Shoup ciphertexts.

Notations. We work in a group $\mathbb{G}$ of prime order p, generated by g, in which we assume the DDH assumption to hold. We define ElGamal encryption scheme with encryption key $\mathsf{ek} = (g, h = g^x)$. We are interested in languages on the ciphertexts $C_{1,i} = (u_{1,i} = g^{r_{1,i}}, e_{1,i} = h^{r_{1,i}} \cdot X_i)$, for $X_1, \ldots, X_{n_1} \in \mathbb{G}$, and $C_{2,j} = (u_{2,j} = g^{r_{2,j}}, e_{2,j} = h^{r_{2,j}} \cdot g^{y_j})$, for $y_1, \ldots, y_{n_2} \in \mathbb{Z}_p$, such that:

$$\prod_{i=1}^{n_1} X_i^{a_i} \cdot \prod_{j=1}^{n_2} A_j^{y_j} = B, \text{ with } \begin{array}{l} \mathsf{crs} = (p, \mathbb{G}, \mathsf{ek}, A_1, \ldots, A_{n_2}) \\ \mathsf{aux} = (a_1, \ldots, a_{n_1}, B) \in \mathbb{Z}_p^{n_1} \times \mathbb{G}. \end{array} \tag{1}$$

We insist that, here, the elements $(A_1, \ldots, A_{n_2}) \in \mathbb{G}^{n_2}$ are known in advance, contrarily to equation (2) in Appendix D.4, where they are in aux and make the SPHF to use pairings.

In the following, i and j will always range from 1 to n_1 and from 1 to n_2 respectively in all the products $\prod_i$, $\prod_j$ and tuples $(\cdot)_i$, $(\cdot)_j$. We can define the following elements, and namely the $(2n_2+1, 2n_2+2)$-matrix Γ that uses the knowledge of the elements $(A_j)_j$:

$$\Gamma = \left(\begin{array}{c|c|c|c} g & 1 \ldots 1 & 1 \ldots 1 & h \\ \hline \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} & \begin{matrix} g & & 1 \\ & \ddots & \\ 1 & & g \end{matrix} & \begin{matrix} h & & 1 \\ & \ddots & \\ 1 & & h \end{matrix} & \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} \\ \hline \begin{matrix} 1 \\ \vdots \\ 1 \end{matrix} & 1 & \begin{matrix} g & & 1 \\ & \ddots & \\ 1 & & g \end{matrix} & \begin{matrix} A_1^{-1} \\ \vdots \\ A_{n_2}^{-1} \end{matrix} \end{array}\right)$$

$$\begin{aligned} \Theta_{\mathsf{aux}}(\boldsymbol{C}) &= \left(\textstyle\prod_i u_{1,i}^{a_i}, (u_{2,j})_j, (e_{2,j})_j, \prod_i e_{1,i}^{a_i}/B\right) \\ \boldsymbol{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, (r_{2,j})_j, (y_j)_j) \\ \boldsymbol{\lambda} \odot \Gamma &= \left(g^{\sum_i a_i r_{1,i}}, (g^{r_{2,j}})_j, (h^{r_{2,j}} \cdot g^{y_j})_j, h^{\sum_i a_i r_{1,i}} / \textstyle\prod_j A_j^{y_j}\right) \end{aligned}$$

We recall that in the matrix, 1 is the neutral element in $\mathbb{G}$ and can thus be ignored. When one considers the discrete logarithms, they become 0, and thus the matrix is triangular. The three diagonal blocks impose the value of $\boldsymbol{\lambda}$, and the last column defines the relation: the last component of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$ is $\prod_i e_{1,i}^{a_i}/B = h^{\sum_i a_i r_{1,i}} \cdot \prod_i X_i^{a_i}/B$, which is equal to the last component of $\boldsymbol{\lambda} \odot \Gamma = h^{\sum_i a_i r_{1,i}} / \prod_j A_j^{y_j}$, if and only if the relation (1) is satisfied. It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $\mathsf{hp}_1 = g^\eta h^\nu$, $(\mathsf{hp}_{2,j} = g^{\theta_j} h^{\mu_j})_j$, and $(\mathsf{hp}_{3,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, (\theta_j)_j, (\mu_j)_j, \nu)$:

$$H = \prod\nolimits_i (u_{1,i}^\eta e_{1,i}^\nu)^{a_i} \cdot \prod\nolimits_j (u_{2,j}^{\theta_j} e_{2,j}^{\mu_j})/B^\nu = \mathsf{hp}_1^{\sum_i a_i r_{1,i}} \cdot \prod\nolimits_j (\mathsf{hp}_{2,j}^{r_{2,j}} \cdot \mathsf{hp}_{3,j}^{y_j}) = H'.$$

As a consequence, the ciphertexts and the projection keys (which have to be exchanged in a protocol) globally consist of $2n_1 + 4n_2 + 1$ elements from $\mathbb{G}$.

Ciphertexts with Randomness Reuse. A first improvement consists in using multiple independent encryption keys for encrypting the y_j's: $\mathsf{ek}_{2,j} = (g, h_{2,j} = g^{x_{2,j}})$, for $j = 1, \ldots, n_2$. This allows to reuse the same random coins [BBS03]. We are interested in languages on the ciphertexts $(C_{1,i} = (u_{1,i} = g^{r_{1,i}}, e_{1,i} = h^{r_{1,i}} \cdot X_i))_i$, for $(X_i)_i \in \mathbb{G}^{n_1}$, with $(r_{1,i})_i \in \mathbb{Z}_p^{n_1}$, and $C_2 = (u_2 = g^{r_2}, (e_{2,j} = h_{2,j}^{r_2} \cdot g^{y_j})_j)$, for $(y_j)_j \in \mathbb{Z}_p^{n_2}$, still satisfying the same relation (1). This improves on the length of the ciphertexts, from $2n_1 + 2n_2$ group elements in $\mathbb{G}$ to $2n_1 + n_2 + 1$. The matrix Γ can then be compressed into:

$$\Gamma = \left(\begin{array}{c|c|ccc|c} g & 1 & 1 & \ldots & 1 & h \\ \hline 1 & g & h_{2,1} & \ldots & h_{2,n_2} & 1 \\ \hline 1 & 1 & g & & 1 & A_1^{-1} \\ \vdots & \vdots & & \ddots & & \vdots \\ 1 & 1 & 1 & & g & A_{n_2}^{-1} \end{array}\right) \qquad \begin{array}{r} \Theta_{\mathsf{aux}}(\boldsymbol{C}) = \left(\prod_i u_{1,i}^{a_i}, u_2, (e_{2,j})_j, \prod_i e_{1,i}^{a_i}/B\right) \\ \boldsymbol{\lambda} = (\sum_i a_i r_{1,i}, r_2, (y_j)_j) \\ \boldsymbol{\lambda} \odot \Gamma = (g^{\sum_i a_i r_{1,i}}, g^{r_2}, (h_{2,j}^{r_2} g^{y_j})_j, h^{\sum_i a_i r_{1,i}} / \prod_j A_j^{y_j}) \end{array}$$

where again, because of the diagonal blocks in Γ, $\boldsymbol{\lambda}$ is implied by all but last components of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$. The last component of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$ is then $\prod_i e_{1,i}^{a_i}/B = \prod_i h^{a_i r_{1,i}} X_i^{a_i}/B$ and thus equal to the last component of $\boldsymbol{\lambda} \odot \Gamma$, multiplied by $\prod_i X_i^{a_i} \cdot \prod_j A_j^{y_j}/B$ that is equal to 1 if and only if the relation (1) is satisfied. It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $(\mathsf{hp}_1 = g^\eta h^\nu$, $\mathsf{hp}_2 = g^\theta \cdot \prod_j h_{2,j}^{\mu_j}$, and $(\mathsf{hp}_{3,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, \theta, (\mu_j)_j, \nu)$:

$$H = \prod\nolimits_i (u_{1,i}^\eta e_{1,i}^\nu)^{a_i} \cdot \prod\nolimits_j e_{2,j}^{\mu_j} \cdot u_2^\eta / B^\nu = \mathsf{hp}_1^{\sum_i a_i r_{1,i}} \cdot \mathsf{hp}_2^{r_2} \cdot \prod\nolimits_j \mathsf{hp}_{3,j}^{y_j} = H'.$$

Globally, the ciphertexts and the projection keys consist of $2n_1 + 2n_2 + 3$ elements from $\mathbb{G}$. This has to be compared with $2n_1 + 4n_2 + 1$ elements from $\mathbb{G}$ in the previous construction.

Moving all the constant values from aux to crs. In some cases, all the constant values, A_j and a_i can be known in advance and public. The matrix Γ can then exploit their knowledge. We apply the randomness-reuse technique for the whole ciphertext, for both $(X_i)_i$ and $(y_j)_j$, with independent encryption keys $(h_{1,i})_i$ and $(h_{2,j})_j$ in $\mathbb{G}$. A unique random r produces $u = g^r$, and $(e_{1,i})_i$ and $(e_{2,j})_j$. This reduces the length of the ciphertext to $n_1 + n_2 + 1$ group elements in $\mathbb{G}$, but also the size of the matrix Γ:

$$\Gamma = \left(\begin{array}{c|ccc|c} g & h_{2,1} & \ldots & h_{2,n_2} & \prod_i h_{1,i}^{a_i} \\ \hline 1 & g & & 1 & A_1^{-1} \\ \vdots & & \ddots & & \vdots \\ 1 & 1 & & g & A_{n_2}^{-1} \end{array}\right) \qquad \begin{array}{r} \Theta_{\mathsf{aux}}(\boldsymbol{C}) = \left(u, (e_{2,j})_j, \prod_i e_{1,i}^{a_i}/B\right) \qquad \boldsymbol{\lambda} = (r, (y_j)_j) \\ \boldsymbol{\lambda} \odot \Gamma = (g^r, (h_{2,j}^r g^{y_j})_j, \prod_i h_{1,i}^{a_i r} / \prod_j A_j^{y_j}) \end{array}$$

Projection keys become more compact, with only $n_2 + 1$ group elements in $\mathbb{G}$: $\mathsf{hp}_1 = g_1^\eta \cdot \prod_j h_{2,j}^{\mu_j} \cdot (\prod_i h_{1,i}^{a_i})^\nu$, and $(\mathsf{hp}_{2,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, (\mu_j)_j, \nu)$: $H = u^\eta \cdot \prod_i e_{1,i}^{\nu a_i} \cdot \prod_j e_{2,j}^{\mu_j} / B^\nu = \mathsf{hp}_1^r \cdot \prod_j \mathsf{hp}_{2,j}^{y_j} = H'$. Globally, the ciphertexts and the projection keys consist of $n_1 + 2n_2 + 2$ elements from $\mathbb{G}$.

4.2 From ElGamal to Cramer-Shoup Encryption

In order to move from ElGamal ciphertexts to Cramer-Shoup ciphertexts, if one already has Γ, Θ_{aux} and $\boldsymbol{\Lambda}$, to guarantee that the ElGamal plaintexts satisfy a relation, one simply has to make a bigger matrix, diagonal per blocks, with blocks Γ and smallers $(\Gamma_k)_k$ for every ciphertext $(u_k, u'_k, e_k, v_k)_k$, where

$$\Gamma_k = \begin{pmatrix} g & 1 & g' & c \\ 1 & g & 1 & d \end{pmatrix} \qquad \boldsymbol{\lambda_k} = (r_k, r_k\xi_k) \qquad \begin{array}{c} \Theta_M(C_k) = (u_k, u_k^{\xi_k}, u'_k, v_k) \\ \boldsymbol{\lambda}_k \odot \Gamma_k = (g^{r_k}, g^{r_k\xi_k}, g'^{r_k}, (cd^{\xi_k})^{r_k}) \end{array}$$

The initial matrix Γ guarantees the relations on the ElGamal pairs (u_k, e_k), and the matrices Γ_k add the internal relations on the Cramer-Shoup ciphertexts. In the worst case, hk is increased by $4n$ scalars and hp by $2n$ group elements, for n ciphertexts. But some more compact matrices can be obtained in many cases, with much shorter hashing and projection keys, by merging some lines or columns in the global matrix. But this is a case by case optimization.

4.3 Generalizations

The SPHF constructions from this section are all done without requiring any pairing, but are still KV-SPHF, allowing us to handle non-quadratic multi-exponentiation equations without pairings. To further extend our formalism, we describe in the next section a concrete application to blind signatures (while with a GL-SPHF), and we present more languages in Appendix D.4.

However, as above for Cramer-Shoup ciphertexts, if one wants to satisfy several equations at a time, one just has to first consider them independently and to make a global matrix with each sub-language-matrix in a block on the diagonal. The hashing keys and the projection keys are then concatenated, and the hash values are simply multiplied. Optimizations can be possible, as shown in Appendix C for the SPHF involved in the blind signature.

4.4 GL-SPHF on Bit Encryption

As shown in Appendix D, our general framework allows to construct KV-SPHFs for any language handled by the Groth-Sahai NIZK proofs. But, while these KV-SPHFs encompass the language of ciphertexts encrypting a bit, they require pairing evaluations. We show here a more efficient GL-SPHF for bit encryption, which does not need pairings.

Let us consider an ElGamal ciphertext $C = (u = g^r, e = h^r g^y)$, in which one wants to prove that $y \in \{0, 1\}$. We can define the following matrix that depends on C, hence a GL-SPHF:

$$\Gamma(C) = \begin{pmatrix} g & h & 1 & 1 \\ 1 & g & u & e/g \\ 1 & 1 & g & h \end{pmatrix} \qquad \begin{array}{c} \Theta_{\mathsf{aux}}(C) = (u, e, 1, 1) \qquad \boldsymbol{\lambda} = (r, y, -ry) \\ \boldsymbol{\lambda} \odot \Gamma(C) = (g^r, h^r g^y, (u/g^r)^y, (e/gh^r)^y) \end{array}$$

Because of the triangular block in $\Gamma(C)$, one sees that $\Theta_{\mathsf{aux}}(C) = \boldsymbol{\lambda} \odot \Gamma(C)$ if and only if $g^{y(y-1)} = 1$, and thus that $y \in \{0, 1\}$. With $\mathsf{hp}_1 = g^\nu h^\theta$, $\mathsf{hp}_2 = g^\theta u^\eta (e/g)^\lambda$, and $\mathsf{hp}_3 = g^\eta h^\lambda$, for $\mathsf{hk} = (\nu, \theta, \eta, \lambda)$: $H = u^\nu e^\theta = \mathsf{hp}_1^r \cdot \mathsf{hp}_2^y / \mathsf{hp}_3^{ry} = H'$.

5 More Applications of SPHFs

5.1 One-Round LAKE

Since we have shown that our framework allows to design KV-SPHFs for complex languages, we extend our PAKE protocol to LAKE [BBC+13]. To this aim, we provide a new security model, inspired from BPR [BPR00] and a complete security proof, which implies the security of our PAKE protocol from Section 2.5.

Review of Language-Authenticated Key Exchange. LAKE is a general framework [BBC+13] that generalizes AKE primitives: each player U owns a word W in a certain language $\mathcal{L}$ and expects the other player to own a word W' in a language $\mathcal{L}'$. If everything is compatible (*i.e.*, the languages are the expected languages and the words are indeed in the appropriate languages), the players compute a common high-entropy secret key, otherwise they learn nothing about the partner's values. In any case, external eavesdroppers do not learn anything, even not the outcome of the protocol: did it succeed or not?

More precisely, we assume the two players have initially agreed on a common public part pub for the languages, but then they secretly parametrize the languages with the private parts priv: $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$ is the language they want to use, and $\mathcal{L}_{\mathsf{pub},priv'}$ is the language they assume the other player will use. In addition, each player owns a word W in his language. We will thus have to use SPHFs on ciphertexts on W, priv and $priv'$, with a common $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$ and aux with the private parameters. For simple languages, this encompasses PAKE and Verifier-based PAKE. We refer to [BBC+13] for more applications of LAKE.

A New Security Model for LAKE. The first security model for LAKE [BBC+13] has been given in the UC framework [Can01], as an extension of the UC security for PAKE [CHK+05]. In this paper, we propose an extension of the PAKE security model presented by Bellare, Pointcheval, and Rogaway [BPR00] model for LAKE: the adversary $\mathcal{A}$ plays a find-then-guess game against n players $(P_i)_{i=1,\dots,n}$. It has access to several instances Π_U^s for each player $U \in \{P_i\}$ and can activate them (in order to model concurrent executions) via several queries: $\mathsf{Execute}$-queries model passive eavesdroppings; Send-queries model active attacks; Reveal-queries model a possible bad later use of the session key; the Test-query models the secrecy of the session key. The latter query has to be asked to a *fresh* instance (which basically means that the session key is not trivially known to the adversary) and models the fact that the session key should look random for an outsider adversary.

Our extension actually differs from the original PAKE security model [BPR00] when defining the quality of an adversary. The goal of an adversary is to distinguish the answer of the Test-query on a fresh instance: a trivial attack is the so-called on-line dictionary attack which consists in trying all the possibilities when interacting with a target player. For PAKE schemes, the advantage of such an attack is q_s/N, where q_s is the number of Send-queries and N the number of possible passwords. A secure PAKE scheme should guarantee this is the best attack, or equivalently that the advantage of any adversary is bounded by $q_s \times 2^{-m}$, where m is the min-entropy of the password distribution. In our extension, for LAKE, the trivial attack consists in trying all the possibilities for $\mathsf{priv}, priv'$ with a word W in $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

Definition 3 (Security for LAKE). *A* LAKE *protocol is claimed* (t, ε)*-secure if the advantage of any adversary running in time* t *is bounded by* $q_s \times 2^{-m} \times \mathsf{Succ}^{\mathcal{L}}(t) + \varepsilon$*, where* m *is the min-entropy of the pair* $(\mathsf{priv}, \mathsf{priv}')$*, and* $\mathsf{Succ}^{\mathcal{L}}(t)$ *is the maximal success an adversary can get in finding a word in any* $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$ *within time* t*.*

Note that the min-entropy of the pair $(\mathsf{priv}, \mathsf{priv}')$ might be conditioned to the public information from the context.

Our Instantiation. Using the same approach as Katz and Vaikuntanathan for their one-round PAKE [KV11], one can design the scheme proposed on Figure 2, in which both users U and U' use the encryption key ek and the public part pub. This defines $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$. When running the protocol, U owns a word W for a private part priv, and thinks about a private part $priv'$ for U', while U' owns a word W' for a private part priv', and thinks about a private $priv$ for U.

This gives a concrete instantiation of one-round LAKE as soon as one can design a KV-SPHF on the language $\textsc{LofC}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},priv')} = \{(\ell, C) \mid \exists r, \exists W, C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, priv', W); r) \text{ and } W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}\}$. More precisely, each player encrypts $(\mathsf{priv}, priv', W)$ as a vector, which thus leads to $C = (C_1, C_2, C_3)$. We then use the combination of three SPHFs: two on equality-test for the

- Players U and U' both use ek and agreed on pub.
- U, with $(\mathsf{priv}, \mathit{priv}', W)$, generates $\mathsf{hk} = \mathsf{HashKG}(\mathsf{ek}, \mathsf{pub})$
  and $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathsf{ek}, \mathsf{pub}), \perp)$.
  U computes $\ell = (U, U', \mathsf{hp})$ and $C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, \mathit{priv}', W); r)$, with r a random scalar in $\mathbb{Z}_p$, and sends hp, C to U'.
- Upon receiving hp', C' from U', it sets $\ell' = (U', U, \mathsf{hp}')$,
  U computes $H = \mathsf{Hash}(\mathsf{hk}, ((\mathsf{ek}, \mathsf{pub}), (\mathit{priv}', \mathsf{priv})), (\ell', C'))$,
  $H' = \mathsf{ProjHash}(\mathsf{hp}', ((\mathsf{ek}, \mathsf{pub}), (\mathsf{priv}, \mathit{priv}')), (\ell, C), r)$, and $\mathsf{sk} = H \cdot H'$.

For $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$ and $\mathsf{aux} = (\mathsf{priv}, \mathit{priv}')$,

$$\mathrm{LofC}_{\mathsf{crs},\mathsf{aux}} = \left\{ (\ell, C) \;\middle|\; \begin{array}{c} \exists r, \exists W, C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, \mathit{priv}', W); r) \\ \text{and } W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}} \end{array} \right\}.$$

Fig. 2. One-Round LAKE

plaintexts priv (for C_1) and priv' (for C_2), and one on $\mathrm{LofC}_{(\mathsf{ek},\mathsf{pub}),\mathsf{priv}}$ for the ciphertext C_3 of $W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

We stress that hk and hp can depend on crs but not on aux, hence the notations used in the Figure 2. Using a similar proof as in [KV11], one can state the following theorem (more details on the security model and the full proof can be found in Appendix B):

Theorem 4. *If the encryption scheme is* IND-CCA*, and* $\mathrm{LofC}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},\mathit{priv}')}$ *languages admit* KV-SPHF*s, then our* LAKE *protocol is secure.*

From LAKE to PAKE. One can remark that this theorem immediately proves the security of our PAKE from Figure 1: one uses $\mathsf{priv} = \mathsf{priv}' = \mathsf{pw}$ and $\mathsf{pub} = \emptyset$, for the language of the ciphertexts of pw.

5.2 Two-Flow Waters Blind Signature

Blind signature schemes, introduced by Chaum in 1982 [Cha83], allow a person to get a signature by another party without revealing any information about the message being signed. A blind signature can then be publicly verified using the unblinded message.

In [BPV12], the authors presented a technique to do efficient blind signatures using an SPHF: it is still the most efficient Waters blind signature known so far. In addition, the resulting signature is a classical Waters signature (see Appendix C.1 for the definition of Waters signatures).

The construction basically consists in encrypting the message bit-by-bit under distinct bases, that will allow the generation of a masked Waters hash of the message. Thereafter, the signer will easily derive a masked signature the user will eventually unmask. However, in order to generate the masked signature, the signer wants some guarantees on the ciphertexts, namely that some ciphertexts contain a bit (in order to allow extractability) and that another ciphertext contains a Diffie-Hellman value. Using our new techniques, we essentially improve on the proof of bit encryption by using the above randomness-reuse technique.

Definition. Before showing our new construction, let us first recall the definition of blind signatures.

A blind signature scheme $\mathcal{BS}$ is defined by three algorithms $(\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSVerif})$ and one interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}, \mathcal{U}\rangle$:

- $\mathsf{BSSetup}(1^{\mathfrak{K}})$, generates the global parameters param of the system;
- $\mathsf{BSKeyGen}(\mathsf{param})$ is a probabilistic polynomial-time algorithm that generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$ where vk is the public (verifying) key and sk is the secret (signing) key;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, M)\rangle$: this is a probabilistic polynomial-time interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk}, M)$, for a message $M \in \{0,1\}^n$. It generates a signature σ on M under vk related to sk for the user.

```
$\mathsf{Exp}^{\mathsf{bl}-b}_{\mathcal{BS},\mathcal{S}^*}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{BSSetup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, M_0, M_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathsf{param})$
3. $\sigma_b \leftarrow \mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, M_b)\rangle$
4. $\sigma_{1-b} \leftarrow \mathsf{BSProtocol}\langle \mathcal{A}, \mathcal{U}(\mathsf{vk}, M_{1-b})\rangle$
5. $b^* \leftarrow \mathcal{S}^*(\mathtt{GUESS} : \sigma_0, \sigma_1)$;
6. RETURN $b^* = b$
```

Blindness property

```
$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\mathfrak{K})$
1. $(\mathsf{param}) \leftarrow \mathsf{BSSetup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{BSKeyGen}(\mathsf{param})$
3. For $i = 1, \dots, q_s$, $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{A}(\mathtt{INIT} : \mathsf{vk})\rangle$
4. $((m_1, \sigma_1), \dots, (m_{q_s+1}, \sigma_{q_s+1})) \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathsf{vk})$;
5. IF $\exists i \neq j, m_i = m_j$ OR $\exists i, \mathsf{Verif}(\mathsf{pk}, m_i, \sigma_i) = 0$ RETURN 0
6. ELSE RETURN 1
```

Unforgeability

Fig. 3. Security Games for $\mathcal{BS}$

- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$ is a deterministic polynomial-time algorithm which outputs 1 if the signature σ is valid with respect to m and vk, 0 otherwise.

A blind signature scheme $\mathcal{BS}$ should satisfy the two following security notions: the blindness condition that is a guarantee for the signer, and the unforgeability that is a guarantee for the signer. The blindness states that a malicious signer should not be able to link the final signatures output by the user to the individual *valid* interactions with the user. We insist on *valid* executions which end with a valid signature σ of the message used by $\mathcal{U}$ under the key vk. The signer could of course send a wrong answer which would lead to an invalid signature in one execution of $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, M)\rangle$. Then, it could easily distinguish a valid signature from an invalid one, and thus valid execution of the protocol and the invalid one. However this malicious behaviour is a kind of denial of service and is out of scope of this work. Therefore, in this paper *blindness* formalizes that one valid execution is indistinguishable for the signer from other valid executions. This notion was formalized in [HKKL07] and termed *a posteriori blindness*. The unforgeability property insures that an adversary, interacting freely with an honest signer, should not be able to produce $q + 1$ valid signatures after at most q complete interactions with the honest signer (for any q polynomial in the security parameter).

These security notions are precised by the security games presented on Figure 3, where the adversary keeps some internal state between the various calls INIT, FIND and GUESS.

Construction. Here, we give a sketch of the protocol (in which i always ranges from 1 to ℓ, except if stated otherwise) and its communication cost:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates a pairing-friendly system $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e; g_1, g_2)$, with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, a random generator $h_s \in \mathbb{G}_1$ as well as independent generators $\boldsymbol{u} = (u_i)_{i \in \{0,\dots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters hash function $\mathcal{F}(M) = u_0 \prod_i u_i^{M_i}$, for $M = (M_i)_i \in \{0,1\}^\ell$, and finally random scalars $(x_i)_i \in \mathbb{Z}_p^\ell$. It also sets $\mathsf{ek} = (h_i)_i = (g_1^{x_i})_i$ and $g_s = \prod_i h_i$. It outputs the global parameters $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \boldsymbol{u})$. Essentially, g_1 and ek compose the encryption key for an ElGamal ciphertext on a vector, applying the randomness-reuse technique, while g_s, g_2 and h_s are the bases used for the Waters signature;
- $\mathsf{KeyGen}(\mathsf{param})$ picks at random $x \in \mathbb{Z}_p$, sets the signing key $\mathsf{sk} = h_s^x$ and the verification key $\mathsf{vk} = (g_s^x, g_2^x)$;
- $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, M)\rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on $M = (M_i)_i \in \{0,1\}^\ell$:
  - Message Encryption: $\mathcal{U}$ chooses a random $r \in \mathbb{Z}_p$ and encrypts $u_i^{M_i}$ for all the i's with the same random r: $c_0 = g_1^r$ and $(c_i = h_i^r u_i^{M_i})_i$. $\mathcal{U}$ also encrypts vk_1^r, into $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$, with a different random s: It eventually sends $(c_0, (c_i)_i, (d_0, d_1)) \in \mathbb{G}_1^{\ell+3}$;
  - Signature Generation: $\mathcal{S}$ first computes the masked Waters hash of the message $c = u_0 \prod_i c_i = (\prod_i h_i)^r \mathcal{F}(M) = g_s^r \mathcal{F}(M)$, and generates the masked signature $(\sigma_1' = h_s^x c^t = h_s^x g_s^{rt} \mathcal{F}(M)^t, \sigma_2 = (g_s^t, g_2^t))$ for a random $t \xleftarrow{\$} \mathbb{Z}_p$;

- SPHF: $\mathcal{S}$ needs the guarantee that each ElGamal ciphertext (c_0, c_i) encrypts either 1 or u_i under the key (g_1, h_i), and (d_0, d_1) encrypts the Diffie-Hellman value of $(g_1, c_0, \mathsf{vk}_1)$ under the key (g_1, h_1). The signer chooses a random $\mathsf{hk} = (\eta, (\theta_i)_i, (\nu_i)_i, \gamma, (\mu_i)_i, \lambda)$ and sets $\mathsf{hp}_1 = g_1^\eta \cdot \prod_i h_i^{\theta_i} \cdot \mathsf{vk}_1^\lambda$, $(\mathsf{hp}_{2,i} = u_i^{\theta_i} c_0^{\nu_i} (c_i/u_i)^{\mu_i})_i$, $(\mathsf{hp}_{3,i} = g_1^{\theta_i} h_i^{\mu_i})_i$, and $\mathsf{hp}_4 = g_1^\gamma h_1^\lambda$, then $H = c_0^\eta \cdot \prod_i c_i^{\theta_i} \cdot d_0^\gamma \cdot d_1^\lambda = \mathsf{hp}_1^r \cdot \prod_i \mathsf{hp}_{2,i}^{M_i} \cdot \mathsf{hp}_{3,i}^{-rM_i} \cdot \mathsf{hp}_4^s = H' \in \mathbb{G}_1$. This SPHF is easily obtained from the above GL-SPHF on bit encryption, as shown in Appendix C;
- Masked Signature: $\mathcal{S}$ sends $(\mathsf{hp}, \Sigma = \sigma_1' \cdot H, \sigma_2) \in \mathbb{G}_1^{2\ell+3} \times \mathbb{G}_2$;
- Signature Recovery: Upon receiving $(\mathsf{hp}, \Sigma, \sigma_2)$, using his witnesses and hp, $\mathcal{U}$ computes H' and unmasks σ_1'. Thanks to the knowledge of r, it can compute $\sigma_1 = \sigma_1' \cdot (\sigma_{2,1})^{-r}$. Note that if $H' = H$, then $\sigma_1 = h_s^x \mathcal{F}(M)^t$, which together with $\sigma_2 = (g_s^t, g_2^t)$ is a valid Waters signature on M;

– $\mathsf{Verif}(\mathsf{vk}, M, (\sigma_1, (\sigma_{2,1}, \sigma_{2,2})))$, checks whether both $e(\sigma_{2,1}, g_2) = e(g_s, \sigma_{2,2})$ and $e(\sigma_1, g_2) = e(h, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2})$ are satisfied or not.

Security Proof. The security proof is similar to the one in [BPV12] and is given in Appendix C.2.

Complexity. The whole process requires only $3\ell + 7$ elements in $\mathbb{G}_1$ ($\ell + 3$ for the ciphertexts, $2\ell + 4$ for the projection key, Σ and $\sigma_{2,1}$) and 1 in $\mathbb{G}_2$ ($\sigma_{2,2}$). This is more efficient than the instantiation from [BPV12] ($5\ell + 6$ elements in $\mathbb{G}_1$ and 1 in $\mathbb{G}_2$) already using an SPHF, and much more efficient than the instantiation from [BFPV11] ($6\ell + 7$ elements in $\mathbb{G}_1$ and $6\ell + 5$ in $\mathbb{G}_2$) using a Groth-Sahai [GS08] NIZK proof.

Acknowledgments

We would like to thank Jonathan Katz for his helpful comments and anonymous referees of Crypto 2013 for their valuable inputs. We also thank Victor Shoup for pointing out to us that the main idea of our generic framework was already present in the full version of [CS02].

This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet. The second author was funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

References

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, August 2009.

BBC+13. Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient uc-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291. Springer, 2013.

BBS03. Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemes. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer, January 2003.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.

BFPV11. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 403–422. Springer, March 2011.

BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000.

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111. Springer, March 2012.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

CCGS10. Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 255–276. Springer, August 2010.

Cha83. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1983.

CHK$^+$05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, April / May 2002.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

GGH12. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610, 2012. `http://eprint.iacr.org/`.

GL03. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

HK08. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, August 2008.

HKKL07. Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 323–341. Springer, February 2007.

JR12. Charanjit S. Jutla and Arnab Roy. Relatively-sound NIZKs and password-based key-exchange. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 485–503. Springer, May 2012.

KOY01. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, May 2001.

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer, March 2011.

LY12. Benoît Libert and Moti Yung. Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 75–93. Springer, March 2012.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*. ACM Press, May 1990.

Poi12. David Pointcheval. Password-based authenticated key exchange (invited talk). In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 390–397. Springer, May 2012.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, October 1999.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.

A Preliminaries

A.1 Formal Definitions of the Basic Primitives

We first recall the definitions of some of the basic tools, with the corresponding security notions and their respective success/advantage.

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions $\mathfrak{H}_K$ from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^{\mathfrak{K}}$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any adversary $\mathcal{A}$ on a random function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\begin{aligned}\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}) &= \Pr[\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(\mathfrak{H}_K) : \mathfrak{H}_K(m_0) = \mathfrak{H}_K(m_1)],\\ \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) &= \max_{\mathcal{A}\leq t}\{\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A})\},\end{aligned}$$

where the latter notation means the maximum over the adversaries running within time t.

Labeled Encryption Scheme. A labeled public-key encryption scheme $\mathcal{E}$ is defined by four algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the encryption key ek and the decryption key dk;
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key ek, using the random coins r;
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, c)$ outputs the plaintext m encrypted in c under the label ℓ, or $\perp$ for an invalid ciphertext.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{ek}, \mathsf{dk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)) = m.$$

- *Indistinguishability under chosen-ciphertext attacks*: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls FIND and GUESS, and makes use of the oracle $\mathsf{ODecrypt}$:
    - $\mathsf{ODecrypt}(\ell, c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key dk. The input queries (ℓ, c) are added to the list $\mathcal{CT}$.

```
$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\ell^*, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek}, \mathsf{ODecrypt}(\cdot,\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\ell^*, \mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot,\cdot))$
6. IF $(\ell^*, c^*) \in \mathcal{CT}$ RETURN 0
7. ELSE RETURN $b'$
```

The advantages are

$$\mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind\text{-}cca}}(\mathcal{A}) = \Pr[\mathsf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind\text{-}cca}-1}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}_{\mathcal{E},\mathcal{A}}^{\mathsf{ind\text{-}cca}-0}(\mathfrak{K}) = 1]$$
$$\mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind\text{-}cca}}(t) = \max_{\mathcal{A}\leq t}\{\mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind\text{-}cca}}(\mathcal{A})\}.$$

A.2 Statistical and Computational Distances

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two probability distributions over a finite set $\mathcal{S}$ and let X and Y be two random variables with these two respective distributions.

Statistical Distance. The statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is also the statistical distance between X and Y:

$$\mathrm{Dist}(\mathcal{D}_1, \mathcal{D}_2) = \mathrm{Dist}(X, Y) = \sum_{x\in\mathcal{S}} |\Pr[X = x] - \Pr[Y = x]|.$$

If the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is less than or equal to ε, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-close or are ε-statistically indistinguishable. If the $\mathcal{D}_1$ and D_2 are 0-close, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are perfectly indistinguishable.

Computational Distance. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are (t, ε)-computationally indistinguishable, if, for every probabilistic algorithm $\mathcal{A}$ running in time at most t:

$$|\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]| \leq \varepsilon.$$

We can note that for any t and ε, $\mathcal{D}_1$ and $\mathcal{D}_2$ are (t, ε)-computationally indistinguishable, if they are ε-close.

A.3 Concrete Instantiations

All the analyses in this paper could be instantiated with ElGamal-like schemes, based on either the Decisional Diffie-Hellman (DDH) assumption, or the Decisional Linear (DLin) assumption. But we focus on the former only:

Definition 5 (Decisional Diffie-Hellman (DDH)). *The Decisional Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b, g^c) for unknown random $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}_{p,\mathbb{G},g}^{\mathsf{ddh}}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple within time t.*

Cramer-Shoup (CS) Encryption Scheme [CS98]: it can be turned into a labeled public-key encryption scheme:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a Collision-Resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/u_1^z$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DDH assumption and the collision-resistance / universal one-wayness of the hash function $\mathcal{H}$.

B Security Proof for LAKE

B.1 Security Model

In this paper, we focus on efficiency and propose (in Section 5.1) an extension of the PAKE security model presented by Bellare-Pointcheval-Rogaway [BPR00] model for PAKE, between n players in the presence of an adversary. The adversary $\mathcal{A}$ plays a find-then-guess game against n players $(P_i)_{i=1,\ldots,n}$. It has access to several instances Π_U^s for each player $U \in \{P_i\}$ and can activate them (in order to model concurrent executions) via several queries, described below:

– $\mathsf{Execute}(U, s, U', t)$: one outputs the transcript of an execution of the protocol between the instance Π_U^s of U and the instance $\Pi_{U'}^t$ of U'. It models passive eavesdropping attacks;
– $\mathsf{Send}(U, s, U', t, m)$: one sends the message m to the instance $\Pi_{U'}^t$ of U' in the name of the instance Π_U^s of U. It models active attacks;
– $\mathsf{Reveal}(U, s)$: if the instance Π_U^s of U has “accepted”, one outputs the session key, otherwise one outputs $\perp$. It models a possible bad later use of the session key;
– $\mathsf{Test}(U, s)$: one first flips a coin $b \overset{\$}{\leftarrow} \{0, 1\}$, if $b = 1$ one outputs $\mathsf{Reveal}(U, s)$, otherwise one outputs a truly random key. It models the secrecy of the session key.

We say that Π_U^s and $\Pi_{U'}^t$ have matching conversations if inputs-outputs of the former correspond to the outputs-inputs of the latter and vice-versa. They are then called *partners*. We say that an instance is *fresh* if the key exists and is not trivially known by the adversary: more precisely, Π_U^s is fresh if

– Π_U^s has *accepted* the session, which is required to compute a session key;
– Π_U^s has not been asked a Reveal-query;
– no $\Pi_{U'}^t$ with *matching conversations* with Π_U^s has been asked a Reveal-query.

A key exchange protocol is then said secure if keys are indistinguishable from random keys for adversaries. Formally, the adversary is allowed to ask as many Execute, Send and Reveal-queries as it likes, and then only one Test-query to a *fresh* instance Π_U^s of a player. The adversary wins if it has guessed correctly the bit b in this query.

B.2 Proof of Theorem 4

This proof follows the one from [KV11]. It starts from the real attack game, in a Game 0: $\mathsf{Adv}_0(\mathcal{A}) = \varepsilon$. We incrementally modify the simulation to make possible the trivial attacks only. In the first games, all the honest players have their own values, and the simulator knows and can use them. Following [KV11], we can assume that there are two kinds of Send-queries: $\mathsf{Send}_0(U, s, U')$-queries where the adversary asks the instance Π_U^s to initiate an execution with an instance of U'. It is answered by the flow U' should send to communicate with U; $\mathsf{Send}_1(U, s, m)$-queries where the adversary sends the message m to the instance Π_U^s. It gives no answer back, but defines the session key, for possible later Reveal or Test-queries.

Game $\mathbf{G}_1$: We first modify the way Execute-queries are answered: we replace C and C' by encryptions of a fixed message M_0, that parses as two private parts P and P' and a word W, such that W is not in the language induced by (pub, P). Since the hashing keys are known, the common session key is computed as

$$\mathsf{sk} = \mathsf{Hash}(\mathsf{hk}, ((\mathsf{ek}, \mathsf{pub}), \mathit{priv}'), C') \times \mathsf{Hash}(\mathsf{hk}', ((\mathsf{ek}, \mathsf{pub}), \mathsf{priv}), C).$$

Since we could have first modified the way to compute sk, that has no impact at all from the soundness of the SPHF, the unique difference comes from the different ciphertexts. This is anyway indistinguishable under the IND-CPA property of the encryption scheme, for each Execute-query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_0(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_2$: We modify again the way $\mathsf{Execute}$-queries are answered: we replace the common session key by a truly random value. Since the languages are not satisfied, the smoothness guarantees indistinguishability: $|\mathsf{Adv}_2(\mathcal{A}) - \mathsf{Adv}_1(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_3$: We now modify the way one answers the Send_1-queries, by using a decryption oracle, or alternatively knowing the decryption key. More precisely, when a message (hp, C) is sent, three cases can appear:

- it has been generated (altered) by the adversary, then one first decrypts the ciphertext to get $(\mathsf{priv}', \mathit{priv}, W')$ used by the adversary. Then
  - If they are correct ($W' \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'}$) and consistent with the receiver's values ($\mathsf{priv}' = \mathit{priv}', \mathit{priv} = \mathsf{priv}$) —event Ev— one declares that $\mathcal{A}$ succeeds (saying that $b' = b$) and terminates the game;
  - if they are not both correct and consistent with the receiver's values, one chooses sk at random.
- it is a replay of a previous flow sent by the simulator, then, in particular, one knows the hashing keys, and one can compute the session keys using all the hashing keys.

The first case can only increase the advantage of the adversary in case Ev happens (which probability is computed in $\mathbf{G}_6$). The second change is indistinguishable under the adaptive-smoothness and thus only increases the advantage of the adversary by a negligible term. The third change does not affect the way the key is computed, so finally: $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{negl}()$.

Game $\mathbf{G}_4$: We modify again the way one answers the Send_1-queries. More precisely, when a message (hp, C) is sent, two cases can appear:

- if there is an instance $\Pi_{U'}^t$ partnered with Π_U^s that receives this flow, then set the key identical to the key for $\Pi_{U'}^t$;
- otherwise, one chooses sk at random.

The former case remains identical since the message is a replay of a previous flow, and the latter is indistinguishable, as in [KV11], thanks to the adaptive-smoothness and their technical lemma that proves that all the hash values are random looking even when hashing keys and ciphertexts are re-used: $|\mathsf{Adv}_4(\mathcal{A}) - \mathsf{Adv}_3(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_5$: We now modify the way one answers the Send_0-queries: instead of encrypting the correct values, one does as in $\mathbf{G}_1$ for $\mathsf{Execute}$-queries, and encrypts M_0. Since for simulating the Send_1-queries decryptions are required, indistinguishability relies on the $\mathsf{IND\text{-}CCA}$ security of the encryption scheme: $|\mathsf{Adv}_5(\mathcal{A}) - \mathsf{Adv}_4(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_6$: For all the hashing and projection keys, we now use the dummy private inputs. Since we restricted hk and hp not to depend on aux, the distributions of these keys are independent of the auxiliary private inputs: $|\mathsf{Adv}_6(\mathcal{A}) - \mathsf{Adv}_5(\mathcal{A})| \leq \mathsf{negl}()$.
If one combines all the relations, one gets $\mathsf{Adv}_6(\mathcal{A}) \geq \mathsf{Adv}_0(\mathcal{A}) - \mathsf{negl}() = \varepsilon - \mathsf{negl}()$.

One can note that in this final game, the values of the honest players are not used anymore during the simulation, but just for declaring whether the adversary has won or not (event Ev). Otherwise, non-partnered players have random and independent keys, and thus unless the simulator stops the simulation, the advantage in the last game is exactly 0: $\mathsf{Adv}_6(\mathcal{A}) = \Pr[\mathsf{Ev}]$. And thus, we have $\varepsilon \leq \Pr[\mathsf{Ev}] + \mathsf{negl}()$.

Let us recall that Ev means that the adversary has encrypted $(\mathsf{priv}', \mathit{priv}, W')$ that are correct ($W' \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'}$) and consistent with the receiver's values ($\mathsf{priv}' = \mathit{priv}', \mathit{priv} = \mathsf{priv}$). Since the values for the honest players are never used during the simulation, we can assume we choose them at the very end only to check whether event Ev happened:

$$\Pr[\mathsf{Ev}] = \Pr[\exists k : \mathsf{priv}'(k) = \mathit{priv}'_{i_k}, \mathit{priv}(k) = \mathit{priv}_{i_k}, W'(k) \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'_{i_k}}]$$

where k lists all the Send_1-queries with adversary-generated messages in which the ciphertexts decrypt to $(\mathsf{priv}'(k), \mathit{priv}(k), W'(k))$, and i_k is the index of the recipient of k-th Send_1-query: it has first to guess the private values, and then once it has guessed them it has to find a word in the language:

$$\Pr[\mathsf{Ev}] \leq \frac{q_s}{2^m} \times \mathsf{Succ}^{\mathrm{L}}(t),$$

where m is the minimal min-entropy on the joint distributions of the $(\mathsf{priv}, \mathit{priv}')$ for any two players U, U' who want to communicate, and $\mathsf{Succ}^{\mathrm{L}}(t)$ is the best success an adversary can get in finding a word in a language $\mathrm{L}_{\mathsf{pub},\mathsf{priv}}$. Then, by combining all the inequalities, one gets

$$\varepsilon \leq \frac{q_s}{2^m} \times \mathsf{Succ}^{\mathrm{L}}(t) + \mathsf{negl}().$$

C Blind Signature

In this appendix, we give details on our two-flow Waters blind signature scheme outlined in Section 5.2. We first present the asymmetric variant of Waters signatures proposed in [BFPV11] and then recall the formal security definitions of blind signatures and of their security properties. Using the formalism from Appendix D, we describe in details the SPHF used in the scheme and finally prove the security of our scheme.

C.1 Waters Signature (Asymmetric Setting)

In 2011, Blazy, Fuchsbauer, Pointcheval and Vergnaud [BFPV11] proposed the following variant of Waters signatures in an asymmetric pairing-friendly environment:

- $\mathsf{Setup}(1^{\mathfrak{K}})$: in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$, one chooses a random vector $\boldsymbol{u} = (u_0, \ldots, u_\ell) \xleftarrow{\$} \mathbb{G}_1^{\ell+1}$, and for convenience, we denote $\mathcal{F}(M) = u_0 \prod_{i=1}^{\ell} u_i^{M_i}$ for $M = (M_i)_i \in \{0,1\}^\ell$. We also need two extra generators $(g_s, h_s) \xleftarrow{\$} \mathbb{G}_1^2$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, g_s, h_s, \boldsymbol{u})$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (g_s^x, g_2^x) = (\mathsf{vk}_1, \mathsf{vk}_2)$, and the secret key is set as $\mathsf{sk} = h_s^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $t \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = \big(\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^t, \sigma_2 = (\sigma_{2,1} = g_s^t, \sigma_{2,2} = g_2^t)\big)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g_2) = e(h_s, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2})$, and $e(\sigma_{2,1}, g_2) = e(g_s, \sigma_{2,2})$.

This scheme is unforgeable against (adaptive) chosen-message attacks under the following variant of the CDH assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$:

Definition 6 (The Advanced Computational Diffie-Hellman problem (CDH^+)). *In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.*

C.2 Underlying SPHF in the Blind Signature Scheme

Following [BPV12], our scheme makes use of an SPHF in the interactive signing protocol to insure (in an efficient way) that the user actually knows the signed message. As outlined in Section 5.2, during the interactive process of the blind signature protocol, we have:

- General setting: a pairing-friendly system $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively;
- Encryption parameters: random scalars $(x_i)_i \in \mathbb{Z}_p^\ell$ with $(h_i = g_1^{x_i})_i$, where i ranges from 1 to ℓ, as everywhere in the following. Then, $\mathsf{ek} = (h_i)_i$;

– Signature parameters: independent generators $\boldsymbol{u} = (u_i)_{i \in \{0,\dots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters function, $g_s = \prod_i h_i$, and a random generator $h_s \in \mathbb{G}_1$, then $\mathsf{sk} = h_s^x$ and $\mathsf{vk} = (g_s^x, g_2^x)$, for a random scalar x.

The user has generated $c_0 = g_1^r$ and $c_i = h_i^r u_i^{M_i}$ for $i = 1, \dots, \ell$, as well as $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$. In the following simulation, we will extract $(M_i)_i$ from $C = (c_0, (c_i)_i)$, and we thus need to be sure that this message can be extracted. In addition, the simulator will also need to know vk_1^r to generate the blinded signature, hence its encryption in (d_0, d_1). But this has to be checked, with the following language membership, where we use notations from Appendix D:

1. each (c_0, c_i) encrypts a bit;

$$\Gamma(C) = \left(\begin{array}{c|c|c|c}
g_1 & h_1 \dots h_\ell & 1 \dots 1 & 1 \dots 1 \\ \hline
\begin{matrix}1\\ \vdots \\ 1\end{matrix} & \begin{matrix} u_1 & & 1 \\ & \ddots & \\ 1 & & u_\ell\end{matrix} & \begin{matrix} c_0 & & 1\\ & \ddots & \\ 1 & & c_0\end{matrix} & \begin{matrix} c_1/u_1 & & 1 \\ & \ddots & \\ 1 & & c_\ell/u_\ell\end{matrix} \\ \hline
\begin{matrix}1\\ \vdots\\ 1\end{matrix} & 1 & \begin{matrix} g_1 & & 1\\ & \ddots & \\ 1 & & g_1\end{matrix} & \begin{matrix} h_1 & & 1\\ & \ddots & \\ 1 & & h_\ell\end{matrix}
\end{array}\right) \quad \begin{aligned} \Theta_{\mathsf{aux}}(C) &= (c_0, (c_i)_i, (1)_i, (1)_i) \\ \boldsymbol{\lambda} &= (r, (M_i)_i, (-rM_i)_i) \end{aligned}$$

$$\boldsymbol{\lambda} \cdot \Gamma(C) = (g_1^r, (h_i^r u_i^{M_i})_i, (c_0^{M_i} g_1^{-rM_i})_i, ((c_i/u_i h_i^r)^{M_i})_i).$$

2. the ciphertext (d_0, d_1) encrypts the Diffie-Hellman value of $(g_1, c_0, \mathsf{vk}_1)$;

$$\Gamma = \begin{pmatrix} g_1 & 1 & \mathsf{vk}_1 \\ 1 & g_1 & h_1 \end{pmatrix} \qquad \begin{aligned} \Theta_{\mathsf{aux}}(C) &= (c_0, d_0, d_1) \\ \boldsymbol{\lambda} \cdot \Gamma &= (g_1^r, g_1^s, \mathsf{vk}_1^r h_1^s) \end{aligned} \qquad \boldsymbol{\lambda} = (r, s)$$

The two matrices can be compressed with a common row/column: the same witness r is indeed used in both matrices, the two corresponding rows can be merged; the first column is the same in both matrices, it can thus be a common one:

$$\Gamma(C) = \left(\begin{array}{c|c|c|c|c|c}
g_1 & h_1 \dots h_\ell & 1 \dots 1 & 1 & 1 \dots 1 & \mathsf{vk}_1 \\ \hline
\begin{matrix}1\\ \vdots \\ 1\end{matrix} & \begin{matrix} u_1 & & 1 \\ & \ddots & \\ 1 & & u_\ell\end{matrix} & \begin{matrix} c_0 & & 1\\ & \ddots & \\ 1 & & c_0\end{matrix} & \begin{matrix}1\\ \vdots \\ 1\end{matrix} & \begin{matrix} c_1/u_1 & & 1 \\ & \ddots & \\ 1 & & c_\ell/u_\ell\end{matrix} & \begin{matrix}1\\ \vdots \\ 1\end{matrix} \\ \hline
\begin{matrix}1\\ \vdots\\ 1\end{matrix} & 1 & \begin{matrix} g_1 & & 1\\ & \ddots & \\ 1 & & g_1\end{matrix} & \begin{matrix}1\\ \vdots \\ 1\end{matrix} & \begin{matrix} h_1 & & 1\\ & \ddots & \\ 1 & & h_\ell\end{matrix} & \begin{matrix}1\\ \vdots \\ 1\end{matrix} \\ \hline
1 & 1 \dots 1 & 1 \dots 1 & g_1 & 1 \dots 1 & h_1
\end{array}\right) \quad \begin{aligned} \Theta_{\mathsf{aux}}(C) &= (c_0, (c_i)_i, (1)_i, d_0, (1)_i, d_1) \\ \boldsymbol{\lambda} &= (r, (M_i)_i, (-rM_i)_i, s) \end{aligned}$$

$$\boldsymbol{\lambda} \cdot \Gamma(C) = (g_1^r, (h_i^r u_i^{M_i})_i, (c_0^{M_i} g_1^{-rM_i})_i, g_1^s, ((c_i/u_i h_i^r)^{M_i})_i, \mathsf{vk}_1^r h_1^s).$$

This leads to, with $\mathsf{hk} = (\eta, \{\theta_i\}_i, \{\nu_i\}_i, \gamma, \{\mu_i\}_i, \lambda)$,

$$\mathsf{hp}_1 = g_1^\eta \cdot \prod\nolimits_i h_i^{\theta_i} \cdot \mathsf{vk}_1^\lambda \qquad (\mathsf{hp}_{2,i} = u_i^{\theta_i} c_0^{\nu_i} (c_i/u_i)^{\mu_i})_i \qquad (\mathsf{hp}_{3,i} = g_1^{\nu_i} h_i^{\mu_i})_i \qquad \mathsf{hp}_4 = g_1^\gamma h_1^\lambda$$

$$H = c_0^\eta \cdot \prod\nolimits_i c_i^{\theta_i} \cdot d_0^\gamma \cdot d_1^\lambda = \mathsf{hp}_1^r \cdot \prod\nolimits_i \mathsf{hp}_{2,i}^{M_i} \cdot \mathsf{hp}_{3,i}^{-rM_i} \cdot \mathsf{hp}_4^s = H'.$$

The signers thus uses H to mask his blinded signature (σ_1', σ_2). But since σ_2 is just a random pair, only σ_1' needs to be masked. Without it, one cannot forge a signature, but it can be unmasked by the user with H', if the values $(c_0, (c_i)_i, (d_0, d_1))$ are in the correct language, and thus are correct ciphertexts.

One can note that the projection key consists of $2\ell + 2$ group elements in $\mathbb{G}_1$, and the hash value is in $\mathbb{G}_1$. No pairings are needed for this SPHF. Since Γ depends on C, this is a GL-SPHF, but this is enough for our interactive protocol.

C.3 Security proofs

Proposition 7. *Our blind signature scheme is blind under the* DDH *assumption in* $\mathbb{G}_1$[2]*:*

$$\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(\mathfrak{K}) \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{DDH}}_{p,\mathbb{G}_1,g_1}(\mathfrak{K}).$$

Proof. Let us consider an adversary $\mathcal{A}$ against the blindness of our scheme. We build an adversary $\mathcal{B}$ against the DDH assumption in $\mathbb{G}_1$.

Game $\mathbf{G}_0$: In a first game $\mathbf{G}_0$, we run the standard protocol:

– $\mathsf{BSSetup}(1^k)$, $\mathcal{B}$ generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. It also generates independent generators $\boldsymbol{u} = (u_i)_{i \in \{0,\dots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters function and sets $\mathsf{ek} = (h_i)_i$ and $g_s = \prod_i h_i$. It generates $h_s = g_s^\alpha \in \mathbb{G}_1$ and defines the global parameters as $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \boldsymbol{u})$;

– The adversary $\mathcal{A}$ generates a verification key $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $e(\mathsf{vk}_1, g_2) = e(g_s, \mathsf{vk}_2)$ and two ℓ-bit messages M^0, M^1.

– $\mathcal{A}$ and $\mathcal{B}$ run twice the interactive issuing protocol, first on the message M^b, and then on the message M^{1-b}:
  - $\mathcal{B}$ chooses a random $r \in \mathbb{Z}_p$ and encrypts $u_i^{M_i}$ for all the i's with the same random r: $c_0 = g_1^r$ and $(c_i = h_i^r u_i^{M_i^b})_i$. $\mathcal{B}$ also encrypts vk_1^r, into $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$ and sends $(c_0, (c_i)_i, (d_0, d_1))$ to $\mathcal{A}$.
  - $\mathcal{A}$ then outputs $(\mathsf{hp}, \Sigma = \sigma_1' \times H, \sigma_2)$
  - $\mathcal{B}$, using its witnesses and hp, computes H' and unmasks $\sigma_1' = \Sigma / H$ which together with σ_2 should be a valid Waters Signature on M^b. It then randomizes the signature with s' to get Σ_b.

  The same is done a second time with M^{1-b} to get Σ_{1-b}.

– $\mathcal{B}$ publishes (Σ_0, Σ_1).

– Eventually, $\mathcal{A}$ outputs b'.

We denote by ε the advantage of $\mathcal{A}$ in this game. By definition, we have:

$$\varepsilon = \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) = \Pr_{\mathbf{G}_0}[b' = 1 | b = 1] - \Pr_{\mathbf{G}_0}[b' = 1 | b = 0] = 2 \times \Pr_{\mathbf{G}_0}[b' = b] - 1.$$

Game $\mathbf{G}_1$: In a second game $\mathbf{G}_1$, we modify the way $\mathcal{B}$ extracts the signatures Σ_b and Σ_{1-b}. Since $\mathcal{B}$ knows the scalar α such that $h_s = g_s^\alpha$ it can compute the secret key $\mathsf{sk} = \mathsf{vk}_1^\alpha$ associated to $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2)$. One can note that, since we focus on valid executions with the signer, and due to the re-randomization of Waters signatures which leads to random signatures, $\mathcal{B}$ can generates itself random signatures on M^b and M^{1-b} using sk. This game is perfectly indistinguishable from the previous one:

$$\Pr_{\mathbf{G}_1}[b' = b] = \Pr_{\mathbf{G}_0}[b' = b].$$

Game $\mathbf{G}_2$: In this final game, we replace all the ciphertexts sent by $\mathcal{B}$ by encryption of random group elements in $\mathbb{G}_1$. For proving indistinguishability with the previous game, we use the hybrid technique for ElGamal ciphertexts with randomness re-use [BBS03]:

$$\varepsilon \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{DDH}}_{p,\mathbb{G}_1,g_1}(\mathfrak{K}) + 2 \times \Pr_{\mathbf{G}_2}[b' = b] - 1.$$

In this last game, the two executions are thus perfectly indistinguishable, and thus $\Pr_{\mathbf{G}_2}[b' = b] = 1/2$ and we get the bound claimed in the proposition. □

[2] This assumption is sometimes referred to as the XDH assumption.

Proposition 8. *Our blind signature scheme is unforgeable under the* CDH^+ *assumption.*

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{A}}(\mathfrak{K}) \leq \Theta\left(\frac{\mathsf{Succ}^{\mathsf{CDH}^+}_{p,\mathbb{G}_1,g_1,\mathbb{G}_2,g_2}(\mathfrak{K})}{q_s\sqrt{\ell}}\right).$$

Proof. Let $\mathcal{A}$ be an adversary against the Unforgeability of the scheme. We assume that this adversary is able after q_s signing queries to output at least $q_s + 1$ valid signatures on different messages (for some q_s polynomial in the security parameter). We now build an adversary $\mathcal{B}$ against the CDH^+ assumption.

- $\mathcal{B}$ is first given a CDH^+ challenge $(g_s, g_2, g_s^x, g_2^x, h_s)$ in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$
- $\mathcal{B}$ emulates $\mathsf{BSSetup}$: it picks a random position $j \overset{\$}{\leftarrow} \{0, \dots, \ell\}$, random indices $y_0, \dots, y_\ell \overset{\$}{\leftarrow} \{0, \dots, 2q_s - 1\}$ and random scalars $z_0, \dots, z_\ell \overset{\$}{\leftarrow} \mathbb{Z}_p$ and publishes $\boldsymbol{u} = (u_i)_{i\in\{0,\dots,\ell\}} \in \mathbb{G}^{\ell+1}$ for the Waters function, where $u_0 = h_s^{y_0-2jq_s} g_s^{z_0}$ and $u_i = h_s^{y_i} g_s^{z_i}$ for $i \in \{1, \dots, \ell\}$. It sets $g_1 = g_s^{\gamma}$ and $\mathsf{ek} = (h_i)_i$ with $h_i = g_1^{a_i} \in \mathbb{G}_1$ for $i \in \{1, \dots, \ell\}$ for some known random scalars $a_1, \dots, a_\ell$ and $\gamma = 1/\sum_i a_i \bmod p$. It keeps secret the associated decryption key $\mathsf{dk} = (a_1, \dots, a_\ell) \in \mathbb{Z}_p^\ell$ and outputs the global $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \boldsymbol{u})$.
- $\mathcal{B}$ then emulates $\mathsf{BSKeyGen}$: it publishes $\mathsf{vk} = (g_s^x, g_2^x)$ from the challenge as its verification key (one can note that recovering the signing key h_s^x is the goal of our adversary $\mathcal{B}$);
- $\mathcal{A}$ can now interact q_s times with the signer, playing the interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}, \mathcal{A}\rangle$:
  - $\mathcal{A}$ sends the bit-per-bit encryptions c_i for $i \in \{1, \dots, \ell\}$, and the extra ciphertext (d_0, d_1) hiding Y the verification key vk_1 raised to the randomness;
  - Thanks to dk, $\mathcal{B}$ is able to extract M from the bit-per-bit ciphertexts (either the decryption leads to u_i and so $M_i = 1$, or to g_1 and so $M_i = 0$), and $Y = \mathsf{vk}_1^r$ from the additional ciphertext (d_0, d_1). One can also compute $c_0^{1/\gamma} = g_s^r$.
  - If one of the extracted terms is not of the right form (either not a bit in the c_i, or $(g_s, g_s^r, \mathsf{vk}_1, Y)$ is not a Diffie-Hellman tuple, which occurs if $e(g_s^r, \mathsf{vk}_2) \neq e(Y, g_2)$ and can thus be checked with a pairing computation), then $\mathcal{A}$ has submitted a "word" not in the appropriate language for the SPHF. Therefore through the smoothness property of the SPHF, it is impossible from a theoretic point of view that the adversary extracts anything from $\mathcal{B}$'s answer, therefore $\mathcal{B}$ simply sends a random element Σ in $\mathbb{G}_1$ together with a valid random pair (g_1^t, g_2^t).
  - If $(g_s, g_s^r, \mathsf{vk}_1, Y)$ is a Diffie-Hellman tuple, one knows that $Y = \mathsf{vk}_1^r$.
    $\mathcal{B}$ computes $H = -2jq_s + y_0 + \sum y_i M_i$ and $J = z_0 + \sum z_i M_i$, $\mathcal{F}(M) = h_s^H g_s^J$. If $H \equiv 0 \bmod p$, it aborts, else it sets

    $$\sigma = (\mathsf{vk}_1^{-J/H} Y^{-1/H} (\mathcal{F}(M) c_0^{1/\gamma})^s, (\mathsf{vk}_1^{-1/H} g_1^s, \mathsf{vk}_2^{-1/H} g_2^s)),$$

    for some random scalar s. Setting $t = s - x/H$, we can see this is indeed a valid signature (as output as the end of the signing interactive protocol), since we have:

    $$\begin{aligned}\sigma_1 &= \mathsf{vk}_1^{-J/H} Y^{-1/H} (\mathcal{F}(M) c_0^{1/\gamma})^s = \mathsf{vk}_1^{-J/H} g_s^{-xr/H} (h_s^H g_s^J g_s^r)^s \\ &= g_s^{-xJ/H} g_s^{-xr/H} (h_s^H g_s^J g_s^r)^t (h_s^H g_s^J g_s^r)^{x/H} = h^x (h^H g_s^J g_s^r)^t \\ &= \mathsf{sk} \cdot \delta^t \qquad \text{where } \delta = \mathcal{F}(M) \times g_s^r \\ \sigma_{2,1} &= \mathsf{vk}_1^{-1/H} g_1^s = g_1^{-x/H} g_1^s = g_1^t \\ \sigma_{2,2} &= \mathsf{vk}_2^{-1/H} g_2^s = g_2^{-x/H} g_2^s = g_2^t\end{aligned}$$

  - $\mathcal{B}$ then acts honestly to send the signature through the SPHF.

After a q_s queries, $\mathcal{A}$ outputs a valid signature σ^* on a new message M^* with non negligible probability.
- As before $\mathcal{B}$ computes $H^* = -2jq_s + y_0 + \sum y_i M_i^*$ and $J^* = z_0 + \sum z_i M_i^*$, $\mathcal{F}(M) = h^{H^*} g_1^{J^*}$.
- If $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ aborts. Otherwise $\sigma^* = (\mathsf{sk} \cdot \mathcal{F}(M^*)^t, g_s^t, g_2^t) = (\mathsf{sk} \cdot g_s^{tJ^*}, g_s^t, g_2^t)$ and so $\sigma_1^*/\sigma_2^{*J^*} = \mathsf{sk} = h_s^x$. Therefore if $\mathcal{A}$'s signature is valid and if $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ solves its CDH^+ challenge.

The probability that all the $H \not\equiv 0 \mod p$ for all the simulations, but $H^* \equiv 0 \mod p$ in the forgery is the $(1, q_s)$-programmability of the Waters function. A full proof showing that it happens with probability in $\Theta(\mathsf{Succ}^{\mathsf{CDH}}_{p,\mathbb{G}_1,g_1,\mathbb{G}_2,g_2}(\mathfrak{K})/q_s\sqrt{\ell})$ can be found in [HK08]. $\square$

D Generic Framework for SPHFs and New Constructions

In this appendix, we introduce our full generic framework for SPHFs using a new notion of graded rings, derived from [GGH12]. It enables to deal with cyclic groups, bilinear groups (with symmetric or asymmetric pairings), or even groups with multi-linear maps. Namely, it handles all the previous constructions from [BBC+13].

Before introducing graded rings and our generic framework, we briefly recall the definition of bilinear groups. The last three subsections are dedicated to instantiations. The last instantiation can deal with any quadratic pairing product equation over ciphertexts, which encompasse all languages handled by Groth-Sahai NIZKs, and so can deal with any NP language. We can see that our generic scheme greatly simplify the construction and the presentation of all the SPHFs presented in these last subsections.

This appendix is very formal and technical. We strongly recommend the reader to first read Sections D.3 and 4 where we give the intuition.

D.1 Bilinear Groups

Let us consider three multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ or $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is called a *bilinear setting* if $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear map (called a pairing) with the following properties:

- *Bilinearity.* For all $(a, b) \in \mathbb{Z}_p^2$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- *Non-degeneracy.* The element $e(g_1, g_2)$ generates $\mathbb{G}_T$;
- *Efficient computability.* The function e is efficiently computable.

It is called a *symmetric* bilinear setting if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. In this case, we denote it $(p, \mathbb{G}, \mathbb{G}_T, e)$ and we suppose $g = g_1 = g_2$. Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, it is called an *asymmetric* bilinear setting one otherwise.

D.2 Graded Rings

Our graded rings are a practical way to manipulate elements of various groups involved with pairings, and more generally, with multi-linear maps. This is a slight variant of the notion of graded encoding proposed in [GGH12], where each element has only one representation, instead of a set of representations, and where we can add two elements even with different indexes.

Indexes Set. As in [GGH12], let us consider a finite set of indexes $\Lambda = \{0, \ldots, \kappa\}^\tau \subset \mathbb{N}^\tau$. In addition to considering the addition law $+$ over Λ, we also consider Λ as a bounded lattice, with the two following laws:

$$\sup(\boldsymbol{v}, \boldsymbol{v}') = (\max(\boldsymbol{v}_1, \boldsymbol{v}'_1), \ldots, \max(\boldsymbol{v}_\tau, \boldsymbol{v}'_\tau)) \qquad \inf(\boldsymbol{v}, \boldsymbol{v}') = (\min(\boldsymbol{v}_1, \boldsymbol{v}'_1), \ldots, \min(\boldsymbol{v}_\tau, \boldsymbol{v}'_\tau)).$$

We also write $\boldsymbol{v} < \boldsymbol{v}'$ (resp. $\boldsymbol{v} \leq \boldsymbol{v}'$) if and only if for all $i \in \{1, \ldots, \tau\}$, $\boldsymbol{v}_i < \boldsymbol{v}'_i$ (resp. $\boldsymbol{v}_i \leq \boldsymbol{v}'_i$). Let $\bar{0} = (0, \ldots, 0)$ and $\top = (\kappa, \ldots, \kappa)$, be the minimal and maximal elements.

Graded Ring. The (κ,τ)-graded ring for a commutative ring R is the set $\mathfrak{G} = \Lambda \times R = \{[\boldsymbol{v}, x] \mid \boldsymbol{v} \in \Lambda, x \in R\}$, where $\Lambda = \{0,\dots,\kappa\}^\tau$, with two binary operations $(+,\cdot)$ defined as follows:

– for every $u_1 = [\boldsymbol{v}_1, x_1], u_2 = [\boldsymbol{v}_2, x_2] \in \mathfrak{G}$: $u_1 + u_2 \overset{\mathsf{def}}{=} [\sup(\boldsymbol{v}_1,\boldsymbol{v}_2), x_1 + x_2]$;
– for every $u_1 = [\boldsymbol{v}_1, x_1], u_2 = [\boldsymbol{v}_2, x_2] \in \mathfrak{G}$: $u_1 \cdot u_2 \overset{\mathsf{def}}{=} [\boldsymbol{v}_1 + \boldsymbol{v}_2, x_1 \cdot x_2]$ if $\boldsymbol{v}_1 + \boldsymbol{v}_2 \in \Lambda$, or $\perp$ otherwise, where $\perp$ means the operation is undefined and cannot be done.

We remark that $\cdot$ is only a partial binary operation and we use the following convention: $\perp + u = u + \perp = u \cdot \perp = \perp \cdot u = \perp$, for any $u \in \mathfrak{G} \cup \{\perp\}$. We then denote $\mathfrak{G}_{\boldsymbol{v}}$ the additive group $\{u = [\boldsymbol{v}', x] \in \mathfrak{G} \mid \boldsymbol{v}' = \boldsymbol{v}\}$. We will make natural use of vector and matrix operations over graded ring elements.

Cyclic Groups and Pairing-Friendly Settings. In the sequel, we consider graded rings over $R = \mathbb{Z}_p$ only, because we will use the vector space structure over $\mathbb{Z}_p$ in the proof of the smoothness of our generic construction of SPHF (see Section D.3). This means we cannot directly deal with constructions in [GGH12] yet. Nevertheless, graded rings enable to easily deal with cyclic groups $\mathbb{G}$ of prime order p, and bilinear groups.

Cyclic Group In this case, $\kappa = \tau = 1$: elements $[0, x]$ of index 0 correspond to scalars $x \in \mathbb{Z}_p$ and elements $[1, x]$ of index 1 correspond to group elements $g^x \in \mathbb{G}$.

Symmetric Bilinear Group. Let $(p, \mathbb{G}, \mathbb{G}_T, e)$ be a symmetric bilinear group, and g be a generator of $\mathbb{G}$. We can represent this bilinear group by a graded ring $\mathfrak{G}$ with $\kappa = 2$ and $\tau = 1$. More precisely, we can consider the following map: $[0, x]$ corresponds to $x \in \mathbb{Z}_p$, $[1, x]$ corresponds to $g^x \in \mathbb{G}$ and $[2, x]$ corresponds to $e(g, g)^x \in \mathbb{G}_T$.

Asymmetric Bilinear Group. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be an asymmetric bilinear group, and g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We can represent this bilinear group by a graded ring $\mathfrak{G}$ with $\kappa = 1$ and $\tau = 2$. More precisely, we can consider the following map: $[(0, 0), x]$ corresponds to $x \in \mathbb{Z}_p$, $[(1, 0), x]$ corresponds to $g_1^x \in \mathbb{G}_1$, $[(0, 1), x]$ corresponds to $g_2^x \in \mathbb{G}_2$ and $[(1, 1), x]$ corresponds to $e(g_1, g_2)^x \in \mathbb{G}_T$.

Notations. We have chosen an additive notation for the group law in $\mathfrak{G}_{\boldsymbol{v}}$. On the one hand, this a lot easier to write generic things done, but, on the other hand, it is a bit cumbersome for bilinear groups to use additive notations. Therefore, when we provide an example with a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, we use multiplicative notation $\cdot$ for the law in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and additive notation $+$ for the law in $\mathbb{Z}_p$, as soon as it is not too complicated. But when needed, we will also use the notation $\oplus$ and $\odot$ which correspond to the addition law and the multiplicative law of the corresponding graded rings. In other words, for any $x, y \in \mathbb{Z}_p$, $u_1, v_1 \in \mathbb{G}_1$, $u_2, v_2 \in \mathbb{G}_2$ and $u_T, v_T \in \mathbb{G}_T$, we have:

$$\begin{aligned} x \oplus y &= x + y & & & x \odot y &= x \cdot y = xy \\ u_1 \oplus v_1 &= u_1 \cdot v_1 = u_1 v_1 & & & x \odot u_1 &= u_1^x \\ u_2 \oplus v_2 &= u_2 \cdot v_2 = u_2 v_2 & & & x \odot u_1 &= u_1^x \\ u_T \oplus v_T &= u_T \cdot v_T & u_1 \odot u_2 &= e(u_1, u_2) & x \odot u_T &= u_T^x. \end{aligned}$$

The element 1 will always denote the neutral element in either $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$ (depending on the context) and not $1 \in \mathbb{Z}_p$, which is not used in our constructions.

D.3 Generic Framework for GL-SPHF/KV-SPHF

In this section, we exhibit a generic framework for SPHF for languages of ciphertexts. This is an extension of the framework described in Section 3 to graded rings. We assume that crs is fixed and we write $\mathrm{L}_{\mathsf{aux}} = \mathrm{LOFC}_{\mathsf{full\text{-}aux}} \subseteq \mathcal{S}et$ where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$.

Language Representation. For a language $\mathrm{L}_{\mathsf{aux}}$, we assume there exist two positive integers k and n, a function $\Gamma : \mathcal{S}et \mapsto \mathfrak{G}^{k\times n}$, and a family of functions $\Theta_{\mathsf{aux}} : \mathcal{S}et \mapsto \mathfrak{G}^{1\times n}$, such that for any word $C \in \mathcal{S}et$, $(C \in \mathrm{L}_{\mathsf{aux}}) \iff (\exists \boldsymbol{\lambda} \in \mathfrak{G}^{1\times k}$ such that $\Theta_{\mathsf{aux}}(C) = \boldsymbol{\lambda} \cdot \Gamma(C))$. If Γ is a constant function (independent of the word C), this defines a KV-SPHF, otherwise this is a GL-SPHF. However, in any case, we need the indexes of the components of $\Gamma(C)$ to be independent of C.

We furthermore require that a user, who knows a witness w of the membership $C \in \mathrm{L}_{\mathsf{aux}}$, can efficiently compute $\boldsymbol{\lambda}$.

Smooth Projective Hash Function. With the above notations, the hashing key is a vector $\mathsf{hk} = \boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\intercal \overset{\$}{\leftarrow} \mathbb{Z}_p^n$, while the projection key is, for a word C, $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \cdot \boldsymbol{\alpha} \in \mathfrak{G}^k$ (if Γ does not depend on C, hp does not depend on C either). Then, the hash value is:

$$H = \mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \stackrel{\mathsf{def}}{=} \quad \Theta_{\mathsf{aux}}(C) \cdot \boldsymbol{\alpha} = \boldsymbol{\lambda} \cdot \boldsymbol{\gamma}(C) \quad \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w) = H'.$$

The set Π of hash values is exactly $\mathfrak{G}_{\boldsymbol{v}_H}$, the set of graded elements of index $\boldsymbol{v}_H$, the maximal index of the elements of $\Theta_{\mathsf{aux}}(C)$.

In addition, the following security analysis proves that the above generic SPHF is perfectly smooth, and thus proves the Theorem 2 as a particular case. We insist that if Γ really depends on C this construction yields a GL-SPHF, whereas when Γ is a constant matrix, we obtain a KV-SPHF, but perfectly smooth in both cases.

Security Analysis. In order to prove the smoothness of the above SPHF, we consider a word $C \notin \mathrm{L}_{\mathsf{aux}}$ and a projection key $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \cdot \boldsymbol{\alpha}$: $\forall \boldsymbol{\lambda} \in \mathfrak{G}^{1\times k}, \Theta_{\mathsf{aux}}(C) \neq \boldsymbol{\lambda} \cdot \Gamma(C)$. Using the projection $\mathfrak{L} : \mathfrak{G} \to \mathbb{Z}_p; u = [\boldsymbol{v}, x] \mapsto x$, which can be seen as the discrete logarithm, and which can be applied component-wise on vectors and matrices, this means that $\mathfrak{L}(\Theta_{\mathsf{aux}}(C))$ is linearly independent from the rows of $\mathfrak{L}(\Gamma(C))$. As a consequence, since $\boldsymbol{\alpha}$ is uniformly random, $\mathfrak{L}(\Theta_{\mathsf{aux}}(C)) \cdot \boldsymbol{\alpha}$ is a random variable independent from $\mathfrak{L}(\boldsymbol{\gamma}(C)) = \mathfrak{L}(\Gamma(C)) \cdot \boldsymbol{\alpha}$, and so from $\mathsf{hp} = \boldsymbol{\gamma}(C)$, since the index of $\boldsymbol{\gamma}(C)$ is a constant and thus $\mathfrak{L}(\boldsymbol{\gamma}(C))$ completely defines $\boldsymbol{\gamma}(C)$. Therefore, H is a uniform element of $\mathfrak{G}_{\boldsymbol{v}_H}$ given hp, aux and C.

D.4 Instantiations

A First Example with Pairings.

Notations. We consider the same kind of equation as in the body of the paper (Section 4.1), but on possibly two different groups $\mathbb{G}_1$ and $\mathbb{G}_2$, of the same prime order p, generated by g_1 and g_2, respectively, with a possible bilinear map into $\mathbb{G}_T$. We assume the DDH assumption hold in both $\mathbb{G}_1$ and $\mathbb{G}_2$. We define ElGamal encryption schemes with encryption keys $\mathsf{ek}_1 = (g_1, h_1 = g_1^{x_1})$ and $\mathsf{ek}_2 = (g_2, h_2 = g_2^{x_2})$ on each group. We are interested in languages on the ciphertexts $C_{1,i} = (u_{1,i} = g_1^{r_{1,i}}, e_{1,i} = h_1^{r_{1,i}} \cdot X_i)$, for $X_1, \dots, X_{n_1} \in \mathbb{G}_1$, and $C_{2,j} = (u_{2,j} = g_2^{r_{2,j}}, e_{2,j} = h_2^{r_{2,j}} \cdot g_2^{y_j})$, for $y_1, \dots, y_{n_2} \in \mathbb{Z}_p$, such that:

$$\prod_{i=1}^{n_1} X_i^{a_i} \cdot \prod_{j=1}^{n_2} A_j^{y_j} = B, \text{ with } \begin{array}{l} \mathsf{crs} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{ek}_1, \mathsf{ek}_2) \\ \mathsf{aux} = (a_1, \dots, a_{n_1}, A_1, \dots, A_{n_2}, B) \in \mathbb{Z}_p^{n_2} \times \mathbb{G}_1^{n_2+1}. \end{array} \qquad (2)$$

We insist that here, contrarily to equation (1) in Section 4.1, the group elements $(A_1, \dots, A_{n_2})$ are part of aux, and thus not known in advance. The matrix Γ cannot depend on them anymore:

$$\Gamma = \left(\begin{array}{c|ccc|c} g_1 & 1 & \dots & 1 & h_1 \\ \hline 1 & g_2 & & 1 & h_2 \\ \vdots & & \ddots & & \vdots \\ 1 & 1 & & g_2 & h_2 \end{array}\right)$$

$$\begin{aligned}
\Theta_{\mathsf{aux}}(\boldsymbol{C}) &= \left(\prod_i u_{1,i}^{a_i}, (e(A_j, u_{2,j}))_j, \prod_i e(e_{1,i}^{a_i}, g_2) \cdot \prod_j e(A_j, e_{2,j})/e(B, g_2)\right)\\
\boldsymbol{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, (A_j^{r_{2,j}})_j)\\
\boldsymbol{\lambda} \cdot \Gamma &= \left(g_1^{\sum_i a_i r_{1,i}}, (e(A_j, g_2^{r_{2,j}}))_j, e(h_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_{2,j}}, h_2)\right)
\end{aligned}$$

We recall that in the matrix, 0 means $[\boldsymbol{v}, 0]$ for the appropriate index $\boldsymbol{v}$, and thus $1_{\mathbb{G}_1} = g_1^0 \in \mathbb{G}_1$ in the first line and column, but $1_{\mathbb{G}_2} = g_2^0 \in \mathbb{G}_2$ in the diagonal block. In addition, in the product $\boldsymbol{\lambda} \cdot \Gamma$, when adding two elements, they are first lifted in the minimal common higher ring, and when multiplying two elements, we either make a simple exponentiation (scalar with a group element) or a pairing (two group elements from different groups).

Because of the diagonal blocks in Γ, $\boldsymbol{\lambda}$ is implied by all but last components of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$, then the last column defines the relation: the last component of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$ is $\prod_i e(h_1^{r_{1,i}a_i} X^{a_i}, g_2) \cdot \prod_j e(A_j, h_2^{r_{2,j}} g_2^{y_j})/e(B, g_2)$, which is equal to the last component of $\boldsymbol{\lambda} \cdot \Gamma$, multiplied by the expression below, that is equal to 1 if and only if the relation (2) is satisfied:

$$\prod_i e(X^{a_i}, g_2) \cdot \prod_j e(A_j, g_2^{y_j})/e(B, g_2) = e\left(\prod_i X^{a_i} \cdot \prod_j A_j^{y_j}/B, g_2\right).$$

It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $\mathsf{hp}_1 = g_1^{\nu} h_1^{\lambda}$ and $(\mathsf{hp}_{2,j} = g_2^{\theta_j} h_2^{\lambda})_j$, for $\mathsf{hk} = (\nu, (\theta_j)_j, \lambda)$:

$$\begin{aligned}
H &= \prod_i e((u_{1,i}^{\nu} e_{1,i}^{\lambda})^{a_i}, g_2) \cdot \prod_j e(A_j, u_{2,j}^{\theta_j} e_{2,j}^{\lambda}) \cdot e(B^{-\lambda}, g_2)\\
&= e(\mathsf{hp}_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_{2,j}}, \mathsf{hp}_{2,j}) = H'.
\end{aligned}$$

As a consequence, the ciphertexts and the projection keys (which have to be exchanged in a protocol) globally consist of $2n_1 + 1$ elements from $\mathbb{G}_1$ and $3n_2$ elements from $\mathbb{G}_2$, and pairings are required for the hash value.

Ciphertexts with Randomness Reuse. We can apply the same improvement as in Section 4.1 by using multiple independent encryption keys in $\mathbb{G}_2$, $\mathsf{ek}_{2,j} = (g_2, h_{2,j} = g_2^{x_{2,j}})$, for $j = 1, \dots, n_2$. This allows to reuse the same random coins [BBS03]. We are interested in languages on the ciphertexts $(C_{1,i} = (u_{1,i} = g_1^{r_{1,i}}, e_{1,i} = h_1^{r_{1,i}} \cdot X_i))_i$, for $(X_i)_i \in \mathbb{G}_1^{n_1}$, with $(r_{1,i})_i \in \mathbb{Z}_p^{n_1}$, and $C_2 = (u_2 = g_2^{r_2}, (e_{2,j} = h_{2,j}^{r_2} \cdot g_2^{y_j})_j)$, for $(y_j)_j \in \mathbb{Z}_p^{n_2}$, with $r_2 \in \mathbb{Z}_p$, still satisfying the same relation (2). This improves on the length of the ciphertexts of the g^{y_i}'s, from $2n_2$ group elements in $\mathbb{G}_2$ to $n_2 + 1$ in $\mathbb{G}_2$. A similar $\mathsf{KV\text{-}SPHF}$ as before can be derived, just modifying the last column vector $(h_2)_j$ by $(h_{2,j})_j$:

$$\Gamma = \left(\begin{array}{c|ccc|c}
g_1 & 1 & \dots & 1 & h_1 \\
\hline
1 & g_2 & & 1 & h_{2,1} \\
\vdots & & \ddots & & \vdots \\
1 & 1 & & g_2 & h_{2,n_2}
\end{array}\right)$$

$$\begin{aligned}
\Theta_{\mathsf{aux}}(\boldsymbol{C}) &= \left(\prod_i u_{1,i}^{a_i}, (e(A_j, u_{2,j}))_j, \prod_i e(e_{1,i}^{a_i}, g_2) \cdot \prod_j e(A_j, e_{2,j})/e(B, g_2)\right)\\
\boldsymbol{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, (A_j^{r_2})_j)\\
\boldsymbol{\lambda} \cdot \Gamma &= \left(g_1^{\sum_i a_i r_{1,i}}, (e(A_j, g_2^{r_2}))_j, e(h_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_2}, h_{2,j})\right)
\end{aligned}$$

It leads to the following $\mathsf{KV\text{-}SPHF}$, with $\mathsf{hp}_1 = g_1^{\nu} h_1^{\lambda}$ and $(\mathsf{hp}_{2,j} = g_2^{\theta_j} h_{2,j}^{\lambda})_j$, for $\mathsf{hk} = (\nu, (\theta_j)_j, \lambda)$:

$$\begin{aligned}
H &= \prod_i e((u_{1,i}^{\nu} e_{1,i}^{\lambda})^{a_i}, g_2) \cdot \prod_j e(A_j, u_{2,j}^{\theta_j} e_{2,j}^{\lambda}) \cdot e(B^{-\lambda}, g_2)\\
&= e(\mathsf{hp}_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_2}, \mathsf{hp}_{2,j}) = H'.
\end{aligned}$$

Globally, the ciphertexts and the projection keys consist of $2n_1+1$ elements from $\mathbb{G}_1$ and $2n_2+1$ elements from $\mathbb{G}_2$, but pairings are still required for the hash value. The prior knowledge of the A_j's allows to avoid pairings, as shown in Section 4.1.

SPHF for Linear Pairing Equations over Ciphertexts. Let us now construct an KV-SPHF for a linear pairing equation in an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$ over ElGamal commitments. This will actually be a particular case of the construction of the next section for quadratic pairing equation. It is thus a warm-up for this more technical instantiation. The construction can obviously be extended to systems of linear pairing equations, and to other commitments schemes using the same methods as in Section 4. It can also be slightly simplified in the case of symmetric bilinear groups.

Notations. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a (asymmetric) bilinear group. Let g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, and let $g_T = e(g_1, g_2)$. Let $\mathsf{ek}_1 = (g_1, h_1 = g_1^{x_1})$, $\mathsf{ek}_2 = (g_2, h_2 = g_2^{x_2})$ and $\mathsf{ek}_T = (g_T, h_T = g_T^{x_T})$ be ElGamal key for encryption scheme in, respectively, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.

We are interested in languages of commitments $(C_{1,i})_i$ of $(X_{1,i})_i \in \mathbb{G}_1^{n_1}$, $(C_{2,j})_j$ of $(X_{2,j})_j \in \mathbb{G}_2^{n_2}$, and $(C_{T,k})_i$ of $(X_{T,k})_k \in \mathbb{G}_T^{n_T}$ such that:

$$\prod_i e(X_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, X_{2,j}) \cdot \prod_k X_{T,k}^{a_{T,k}} = B, \tag{3}$$

with $\mathsf{aux} = ((A_{1,j})_j, (A_{2,i})_i, (a_{T,k})_k) \in \mathbb{G}_1^{n_2} \times \mathbb{G}_2^{n_1} \times \mathbb{Z}_p^{n_T}$. This can also be written:

$$\left(\bigoplus_{i=1}^{n_1} A_{2,i} \odot X_{1,i}\right) \oplus \left(\bigoplus_{j=1}^{n_2} A_{1,j} \odot X_{2,j}\right) \oplus \left(\bigoplus_{k=1}^{n_T} a_{T,k} \odot X_{T,k}\right) = B.$$

Let us also write, for any $\omega \in \{1, 2, T\}$ and $\iota \in \{1, \dots, n_\omega\}$: $C_{\omega,\iota} = (u_{\omega,\iota} = g_\omega^{r_{\omega,\iota}}, e_{\omega,\iota} = h_\omega^{r_{\omega,\iota}} X_{\omega,\iota})$. Words of $\mathcal{S}et$ are tuple $C = (C_{\omega,\iota})_{\omega\in\{1,2,T\},\, \iota\in\{1,\dots,n_\omega\}}$.

Basic Scheme in $\mathbb{G}_T$. Let us consider

$$\Gamma = \begin{pmatrix} g_1 & 1 & 1 & h_1 \\ 1 & g_2 & 1 & h_2 \\ 1 & 1 & g_T & h_T \end{pmatrix}$$
$$\Theta(C) = \begin{pmatrix} \bigoplus_i A_{2,i} \odot u_{1,i}, \bigoplus_j A_{1,j} \odot u_{2,j}, \bigoplus_k a_{T,k} \odot u_{T,k}, \\ (\bigoplus_i A_{2,i} \odot e_{1,i}) \oplus \left(\bigoplus_j A_{1,j} \odot e_{2,j}\right) \oplus (\bigoplus_k a_{T,k} \odot e_{T,k}) \ominus B \end{pmatrix}.$$

Because of the diagonal block in Γ, one can note that the unique possibility is

$$\boldsymbol{\lambda} = (\bigoplus_i A_{2,i} \odot r_{1,i}, \bigoplus_j A_{1,j} \odot r_{2,j}, \bigoplus_k r_{T,k}) = (\prod_i A_{2,i}^{r_{1,i}}, \prod_j A_{1,j}^{r_{2,j}}, \sum_k r_{T,k}).$$

We then have $\boldsymbol{\lambda} \odot \Gamma = \Theta(C)$ if and only if

$$\prod_i e(h_1^{r_{1,i}}, A_{2,i}) \cdot \prod_j e(A_{1,j}, h_2^{r_{2,j}}) \cdot \prod_k h_T^{r_{T,k}} = \prod_i e(e_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, e_{2,j}) \cdot \prod_k e_{T,k}^{a_{T,k}} / B$$

and thus if and only if Equation (3) is true, *i.e.*, the word is in the language. Furthermore, if we set $\gamma_1 = g_1^{\alpha_1} h_1^{\alpha^4}$, $\gamma_2 = g_2^{\alpha_2} h_2^{\alpha^4}$, and $\gamma_3 = g_T^{\alpha_3} h_T^{\alpha^4}$, we have

$$\begin{aligned}H &= \left(\prod_{i=1}^{n_1} e(u_{1,i}, A_{2,i})\right)^{\alpha_1} \cdot \left(\prod_{j=1}^{n_2} e(A_{1,j}, u_{2,j})\right)^{\alpha_2} \cdot \left(\prod_{k=1}^{n_T} u_{T,k}^{a_{T,k}}\right)^{\alpha_3} \\ &\times \left(\prod_{i=1}^{n_1} e(e_{1,i}, A_{2,i}) \cdot \prod_{j=1}^{n_2} e(A_{1,j}, e_{2,j}) \cdot \prod_{k=1}^{n_T} e_{T,k}^{a_{T,k}}/B\right)^{\alpha_4} \\ &= e(\gamma_1, \textstyle\prod_i A_{2,i}^{r_{1,i}}) \cdot e(\prod_j A_{1,j}^{r_{2,j}}, \gamma_2) \cdot \gamma_3^{\sum_k r_{T,k}} \qquad = H'.\end{aligned}$$

Variant. The above scheme is not efficient enough for practical use because elements in $\mathbb{G}_T$ are often big and operations in $\mathbb{G}_T$ are often slow. If $h_T = e(h_1, g_2)$, then the last row of Γ can be $(0, 0, g_1, h_1)$ which enables faster hashing and shorter projection key. We remark this modified encryption scheme in $\mathbb{G}_T$ is IND-CPA as soon as DDH is hard in $\mathbb{G}_1$, which we need to suppose for the ElGamal encryption scheme in $\mathbb{G}_1$ to be IND-CPA. So this variant is always more efficient when using ElGamal encryption.

However, if DDH is easy, as in symmetric bilinear group, this variant may not be interesting, since it requires to use the linear encryption scheme in $\mathbb{G}_T$ instead of the ElGamal one.

SPHF for Quadratic Pairing Equations over Ciphertexts. In this section, we present a KV-SPHF for language of ElGamal commitments verifying a quadratic pairing equation. As usual, it can be extended to systems of quadratic pairing equations, and to other commitments schemes. We use the same notations as in the previous construction.

Example. Before showing the generic construction, we describe it on a simple example: we are interested in languages of the ciphertexts $C_1 = (u_1 = g_1^{r_1}, e_1 = h_1^{r_1} X_1)$ and $C_2 = (u_2 = g_2^{r_2}, e_2 = h_2^{r_2} X_2)$, that encrypt two values X_1 and X_2 such that $e(X_1, X_2) = B$ where B is some constant in $\mathbb{G}_T$ and $\mathsf{aux} = B$. We remark the equation $e(X_1, X_2) = B$ can also be written $X_1 \odot X_2 = B$. Let us consider

$$\Gamma = \begin{pmatrix} g_1 \odot g_2 & 1 & 1 & h_1 \odot h_2 \\ 1 & g_1 & 1 & h_1 \\ 1 & 1 & g_2 & h_2 \end{pmatrix} \qquad \begin{aligned}\Theta(C) &= (-u_1 \odot u_2, u_1 \odot e_2, e_1 \odot u_2, e_1 \odot e_2 \ominus B) \\ &= (e(u_1, u_2)^{-1}, e(u_1, e_2), e(e_1, u_2), e(e_1, e_2)/B).\end{aligned}$$

Because of the diagonal block in Γ, one can note that the unique possibility is

$$\boldsymbol{\lambda} = (-r_1 r_2, r_1 \odot e_2, r_2 \odot e_1) = (-r_1 r_2, e_2^{r_1}, e_1^{r_2}).$$

We have $\boldsymbol{\lambda} \odot \Gamma = \Theta(C)$ if and only if $e(h_1, h_2)^{-r_1 r_2} \cdot e(h_1, e_2^{r_1}) \cdot e(e_1^{r_2}, h_2) = e(e_1, e_2)/B$, and thus,

$$\begin{aligned}B &= e(e_1, e_2)/(e(h_1^{r_1}, X_2) \cdot e(e_1, h_2^{r_2})) \\ &= e(e_1, X_2)/e(h_1^{r_1}, X_2) = e(X_1, X_2)\end{aligned}$$

For the sake of completeness, if $\gamma_1 = e(g_1, g_2)^{\alpha_1} e(h_1, h_2)^{\alpha_4}$, $\gamma_2 = g_1^{\alpha_2} h_1^{\alpha_4}$, and $\gamma_3 = g_2^{\alpha_3} h_2^{\alpha_4}$, the corresponding hash value is:

$$H = e(u_1, u_2)^{-\alpha_1} \cdot e(u_1, e_2)^{\alpha_2} \cdot e(e_1, u_2)^{\alpha_3} \cdot (e(e_1, e_2)/B)^{\alpha_4} = \gamma_1^{-r_1 r_2} \cdot e(\gamma_2, e_2^{r_1}) \cdot e(e_1^{r_2}, \gamma_3).$$

Notations. Let us now introduce notation to handle any quadratic equation. In addition to previous notations, as in Section D.4, we also write $\mathsf{ek}_T = (g_T, h_T = g_T^{x_T})$ a public key for ElGamal encryption scheme in $\mathbb{G}_T$. We are interested in languages of commitments $(C_{1,i})_i$ of $(X_{1,i})_i \in \mathbb{G}_1^{n_1}$, $(C_{2,j})_j$ of $(X_{2,j})_j \in \mathbb{G}_2^{n_2}$, and $(C_{T,k})_i$ of $(X_{T,k})_k \in \mathbb{G}_T^{n_T}$ such that:

$$\prod_i e(X_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, X_{2,j}) \cdot \prod_i \prod_j e(X_{1,i}, X_{2,j})^{a_{i,j}} \cdot \prod_k X_{T,k}^{a_{T,k}} = B, \tag{4}$$

with $\mathsf{aux} = ((A_{2,i})_i, (A_{1,j})_j, (a_{i,j})_{i,j}, (a_{T,k})_k) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{Z}_p^{n_1 n_2 + n_T}$. This can also be written:

$$\left(\bigoplus_{i=1}^{n_1} A_{2,i} \odot X_{1,i}\right) \oplus \left(\bigoplus_{j=1}^{n_2} A_{1,j} \odot X_{2,j}\right) \oplus \left(\bigoplus_{i=1}^{n_1}\bigoplus_{j=1}^{n_2} a_{i,j} \odot X_{1,i} \odot X_{2,j}\right) \oplus \left(\bigoplus_{k=1}^{n_T} a_{T,k} \odot X_{T,k}\right) = B.$$

Let us also write, for any $\omega \in \{1,2,T\}$ and $\iota \in \{1,\dots,n_\omega\}$: $C_{\omega,\iota} = (u_{\omega,\iota} = g_\omega^{r_{\omega,\iota}}, e_{\omega,\iota} = h_\omega^{r_{\omega,\iota}} X_{\omega,\iota})$.

Basic Scheme in $\mathbb{G}_T$. Let us consider the following matrix, with a diagonal block

$$\Gamma = \begin{pmatrix} g_1 \odot g_2 & 1 & 1 & 1 & h_1 \odot h_2 \\ 1 & g_1 & 1 & 1 & h_1 \\ 1 & 1 & g_2 & 1 & h_2 \\ 1 & 1 & 1 & g_T & h_T \end{pmatrix}$$

With $\Theta(C)$ equals to

$$\begin{pmatrix} \bigoplus_i \bigoplus_j -a_{i,j} \odot u_{1,i} \odot u_{2,j}, \left(\bigoplus_i \bigoplus_j a_{i,j} \odot u_{1,i} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot u_{1,i}\right), \\ \left(\bigoplus_i \bigoplus_j a_{i,j} \odot e_{1,i} \odot u_{2,j}\right) \oplus \left(\bigoplus_j A_{1,j} \odot u_{2,j}\right), \bigoplus_i a_{T,i} \odot u_{T,i}, \\ \left(\bigoplus_i \bigoplus_j a_{i,j} \odot e_{2,i} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot e_{1,i}\right) \oplus \left(\bigoplus_j A_{1,j} \odot e_{2,j}\right) \oplus \left(\bigoplus_k a_{T,k} \odot e_{T,k}\right) \ominus B \end{pmatrix}$$

the requirement $\boldsymbol{\lambda} \odot \Gamma = \Theta(C)$ implies

$$\begin{aligned} \boldsymbol{\lambda} &= \begin{pmatrix} \bigoplus_i \bigoplus_j -a_{i,j} \odot r_{1,i} \odot r_{2,j}, \left(\bigoplus_i \bigoplus_j r_{1,i} \odot a_{i,j} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot r_{1,i}\right), \\ \left(\bigoplus_i \bigoplus_j r_{2,i} \odot a_{i,j} \odot e_{1,j}\right) \oplus \left(\bigoplus_j A_{1,j} \odot r_{2,j}\right), \bigoplus_k r_{T,k} \end{pmatrix} \\ &= \left(\sum_i \sum_j a_{i,j} r_{1,i} r_{2,j}, \prod_i \prod_j e_{2,j}^{r_{1,i} a_{i,j}} \cdot \prod_i A_{2,i}^{r_{1,i}}, \prod_i \prod_j e_{1,j}^{r_{2,i} a_{i,j}} \cdot \prod_j A_{1,j}^{r_{2,j}}, \sum_k r_{T,k}\right), \end{aligned}$$

and it is satisfied, if and only if Equation (4) is true, *i.e.*, the word is in the language.

Variant. The same trick as the one used in the variant of the SPHF for linear pairing equation can be used to avoid having too many elements of the projection key in $\mathbb{G}_T$.

Appendix C

SPHF-Friendly Non-Interactive Commitments [ABB+13]

This is the Full Version of the Extended Abstract that appears in Advances in Cryptology — Proceedings of ASIACRYPT'2013 (1 – 5 December 2013, Bangalore, India), Kazue Sako and Palash Sarkar Eds., Springer-Verlag, Part I, LNCS 8269, pages 214–234.

Authors

Michel Abdalla, Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval

Abstract

In 2009, Abdalla *et al.* proposed a reasonably practical password-authenticated key exchange (PAKE) secure against adaptive adversaries in the universal composability (UC) framework. It exploited the Canetti-Fischlin methodology for commitments and the Cramer-Shoup smooth projective hash functions (SPHFs), following the Gennaro-Lindell approach for PAKE. In this paper, we revisit the notion of non-interactive commitments, with a new formalism that implies UC security. In addition, we provide a quite efficient instantiation. We then extend our formalism to SPHF-friendly commitments. We thereafter show that it allows a blackbox application to one-round PAKE and oblivious transfer (OT), still secure in the UC framework against adaptive adversaries, assuming reliable erasures and a single global common reference string, even for multiple sessions. Our instantiations are more efficient than the Abdalla *et al.* PAKE in Crypto 2009 and the recent OT protocol proposed by Choi *et al.* in PKC 2013. Furthermore, the new PAKE instantiation is the first one-round scheme achieving UC security against adaptive adversaries.

1 Introduction

Commitment schemes are one of the most fundamental primitives in cryptography, serving as a building block for many cryptographic applications such as zero-knowledge proofs [GMW91] and secure multi-party computation [GMW87]. In a typical commitment scheme, there are two main phases. In a *commit* phase, the committer computes a commitment C for some message x and sends it to the receiver. Then, in an *opening* phase, the committer releases some information δ to the receiver which allows the latter to verify that C was indeed a commitment of x. To be useful in practice, a commitment scheme should satisfy two basic security properties. The first one is *hiding*, which informally guarantees that no information about x is leaked through the commitment C. The second one is *binding*, which guarantees that the committer cannot generate a commitment C that can be successfully opened to two different messages.

Smooth Projective Hash Functions (SPHFs) were introduced by Cramer and Shoup [CS02] as a means to design chosen-ciphertext-secure public-key encryption schemes. In addition to providing a more intuitive abstraction for their original public-key encryption scheme in [CS98], the notion of SPHF also enabled new efficient instantiations of their scheme under different complexity assumptions, such as quadratic residuosity. Due to its usefulness, the notion of SPHF was later extended to several other contexts, such as password-authenticated key exchange (PAKE) [GL03], oblivious transfer (OT) [Kal05,CKWZ13], and blind signatures [BPV12,BBC+13].

Password-Authenticated Key Exchange (PAKE) protocols were proposed in 1992 by Bellovin and Merritt [BM92] where authentication is done using a simple password, possibly drawn from a small space subject to exhaustive search. Since then, many schemes have been proposed and studied. SPHFs have been extensively used, starting with the work of Gennaro and Lindell [GL03] which generalized an earlier construction by Katz, Ostrovsky, and Yung (KOY) [KOY01], and followed by several other works [CHK+05,ACP09]. More recently, a variant of SPHFs proposed by Katz and Vaikuntanathan even allowed the construction of one-round PAKE schemes [KV11,BBC+13].

The first ideal functionality for PAKE protocols in the UC framework [Can01,CK02] was proposed by Canetti *et al.* [CHK+05], who showed how a simple variant of the Gennaro-Lindell methodology [GL03] could lead to a secure protocol. Though quite efficient, their protocol was not known to be secure against adaptive adversaries, that are capable of corrupting players at any time, and learn their internal states. The first ones to propose an adaptively secure PAKE in the UC framework were Barak *et al.* [BCL+05] using general techniques from multi-party computation (MPC). Though conceptually simple, their solution results in quite inefficient schemes.

The first reasonably practical adaptively secure PAKE was proposed by Abdalla *et al.* [ACP09], following the Gennaro-Lindell methodology with the Canetti-Fischlin commitment [CF01]. They had to build a complex SPHF to handle the verification of such a commitment. Thus, the communication complexity was high and the protocol required four rounds. No better adaptively secure scheme has been proposed so far.

Oblivious Transfer (OT) was introduced in 1981 by Rabin [Rab81] as a way to allow a receiver to get exactly one out of k messages sent by another party, the sender. In these schemes, the receiver should be oblivious to the other values, and the sender should be oblivious to which value was received. Since then, several instantiations and optimizations of such protocols have appeared in the literature, including proposals in the UC framework [NP01,CLOS02].

More recently, new instantiations have been proposed, trying to reach round-optimality [HK07], or low communication costs [PVW08]. The 1-out-of-2 OT scheme by Choi *et al.* [CKWZ13] based on the DDH assumption seems to be the most efficient one among those that are secure against adaptive corruptions in the CRS model with erasures. But it does not scale to 1-out-of-k OT, for $k > 2$.

1.1 Properties of Commitment Schemes

Basic Properties. In addition to the binding and hiding properties, certain applications may require additional properties from a commitment scheme. One such property is *equivocability* [Bea96], which guarantees that a commitment C can be opened in more than a single way when in possession of a certain trapdoor information. Another one is *extractability*, which allows the computation of the message x committed in C when in possession of a certain trapdoor information. Yet another property that may also be useful for cryptographic applications is *non-malleability* [DDN00], which ensures that the receiver of a unopened commitment C for a message x cannot generate a commitment for a message that is related to x.

Though commitment schemes satisfying stronger properties such as *non-malleability*, *equivocability*, and *extractability* may be useful for solving specific problems, they usually stop short of guaranteeing security when composed with arbitrary protocols. To address this problem, Canetti and Fischlin [CF01] proposed an ideal functionality for commitment schemes in the universal composability (UC) framework [Can01] which guarantees all these properties simultaneously and remain secure even under concurrent compositions with arbitrary protocols. Unfortunately, they also showed that such commitment schemes can only be realized if one makes additional setup assumptions, such as the existence of a common reference string (CRS) [CF01], random oracles [HMQ04], or secure hardware tokens [Kat07].

Equivocable and Extractable Commitments. As the work of Canetti and Fischlin [CF01], this work also aims to build *non-interactive* commitment schemes which can simultaneously guarantee *non-malleability*, *equivocability*, and *extractability* properties. To this end, we first define a new notion of commitment scheme, called $\mathtt{E}^2$-commitments, for which there exists an alternative setup algorithm, whose output is computationally indistinguishable from that of a normal setup algorithm and which outputs a common trapdoor that allows for both equivocability and extractability: this trapdoor not only allows for the extraction of a committed message, but it can also be used to create simulated commitments which can be opened to any message.

To define the security of $\mathtt{E}^2$-schemes, we first extend the security notions of standard equivocable commitments and extractable commitments to the $\mathtt{E}^2$-commitment setting: Since the use of a common trapdoor for equivocability and extractability could potentially be exploited by an adversary to break the extractability or equivocability properties of an $\mathtt{E}^2$-commitment scheme, we define stronger versions of these notions, which account for the fact that the same trapdoor is used for both extractability or equivocability. In particular, in these stronger notions, the adversary is given oracle access to the simulated commitment and extractor algorithms.

Finally, after defining the security of $\mathtt{E}^2$-schemes, we further show that these schemes remain secure even under arbitrary composition with other cryptographic protocols. More precisely, we show that any $\mathtt{E}^2$–commitment scheme which meets the strong versions of the equivocability or extraction notions is a non-interactive UC-secure (multiple) commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and a single global CRS.

SPHF-Friendly Commitments. In this work, we are interested in building non-interactive $\mathtt{E}^2$-commitments, to which smooth projective hash functions can be efficiently associated. Unfortunately, achieving this goal is not so easy due to the equivocability property of $\mathtt{E}^2$-commitments. To understand why, let X be the domain of an SPHF function and let L be some underlying NP language such that it is computationally hard to distinguish a random element in L from a random element in $X \setminus L$. A key property of these SPHF functions that makes them so useful for applications such as PAKE and OT is that, for words C in L, their values can be computed using either a *secret* hashing key hk or a *public* projected key hp together a witness w to the fact that C is indeed in L. A typical example of a language in which we are interested is the language L_x corresponding to the set of elements $\{C\}$ such that C is a valid commitment of x. Unfortunately, when commitments are equivocable, the language L_x containing the set of valid commitments of x may not be well defined since a commitment C could potentially be opened to any x. To get around this problem and be able to use SPHFs with $\mathtt{E}^2$-commitments, we show that it suffices for an $\mathtt{E}^2$-commitment scheme to satisfy two properties. The first one is the stronger version of the equivocability notion, which guarantees that equivocable commitments are computationally indistinguishable from normal commitments, even when given oracle access to the simulated commitment and extractor algorithms. The second one, which is called *robustness*, is new and guarantees that commitments generated by polynomially-bounded adversaries are perfectly binding. Finally, we say that a commitment scheme is *SPHF-friendly* if it satisfies both properties and if it admits an SPHF on the languages L_x.

1.2 Contributions

A new SPHF-friendly $\mathtt{E}^2$-commitment construction. First, we define the notion of SPHF-friendly $\mathtt{E}^2$-commitment together with an instantiation. The new construction, which is called $\mathcal{E}^2\mathcal{C}$ and described in Section 4, is inspired by the commitment schemes in [CF01,CLOS02,ACP09]. Like the construction in [ACP09], it combines a variant of the Cramer-Shoup encryption scheme (as an extractable commitment scheme) and an equivocable commitment scheme to be able to simultaneously achieve both equivocability and extractability. However, unlike the construction in [ACP09], we rely on Haralambiev's perfectly hiding commitment [Har11, Section 4.1.4], instead of the Pedersen commitment [Ped92].

Since the opening value of Haralambiev's scheme is a group element that can be encrypted in one ElGamal-like ciphertext to allow extractability, this globally leads to a better communication and computational complexity for the commitment. The former is linear in $m \cdot \mathfrak{K}$, where m is the bit-length of the committed value and $\mathfrak{K}$, the security parameter. This is significantly better than the extractable commitment construction in [ACP09] which was linear in $m \cdot \mathfrak{K}^2$, but asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m. However, we point out the latter proposals in [FLM11] are not SPHF-friendly since they are not robust.

We then show in Theorem 4 that a labeled $\mathtt{E}^2$-commitment satisfying stronger notions of equivocability and extractability is a non-interactive UC-secure commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and a single global CRS, and we apply this result to our new construction.

One-round adaptively secure PAKE. Second, we provide a generic construction of a one-round UC-secure PAKE from any SPHF-friendly commitment, verifying an additional property called strong pseudo-randomness. The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS, as shown in Section 6. In addition to being the first one-round adaptively secure PAKE, our new scheme also enjoys a much better communication complexity than previous adaptively secure PAKE schemes. For instance, in comparison to the PAKE in [ACP09], which is currently the most efficient adaptively secure PAKE, the new scheme gains a factor of $\mathfrak{K}$ in the overall communication complexity, where $\mathfrak{K}$ is the security parameter. However, unlike their scheme, our new construction requires pairing-friendly groups.

Three-round adaptively secure 1-out-of-k OT. Third, we provide a generic construction of a three-round UC-secure 1-out-of-k OT from any SPHF-friendly commitment. The UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS, as shown in Section 7. Besides decreasing the total number of rounds with respect to existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solution so far [CKWZ13]. Moreover, our construction is more general and provides a solution for 1-out-of-k OT schemes while the solution in [CKWZ13] only works for $k = 2$.

Due to space restrictions, complete proofs and some details were postponed to the Appendix.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathtt{hid}\text{-}b}(\mathfrak{K})$
    $\rho \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}})$
    $(\ell, x_0, x_1, \mathsf{state}) \overset{\$}{\leftarrow} \mathcal{A}(\rho)$
    $(C, \delta) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(x_b)$
    return $\mathcal{A}(\mathsf{state}, C)$
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathtt{bind}}(\mathfrak{K})$
    $\rho \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}})$
    $(C, \ell, x_0, \delta_0, x_1, \delta_1) \overset{\$}{\leftarrow} \mathcal{A}(\rho)$
    if $\neg\mathsf{VerCom}^{\ell}(C, x_0, \delta_0)$ then return $0$
    if $\neg\mathsf{VerCom}^{\ell}(C, x_1, \delta_1)$ then return $0$
    return $x_0 \neq x_1$
```

Fig. 1. Hiding and Binding Properties

2 Basic Notions for Commitments

We first review the basic definitions of non-interactive commitments, with some examples. Then, we consider the classical additional notions of equivocability and extractability. In this paper, the qualities of adversaries will be measured by their successes and advantages in certain experiments $\mathsf{Exp}^{\mathsf{sec}}$ or $\mathsf{Exp}^{\mathsf{sec}\text{-}b}$ (between the cases $b = 0$ and $b = 1$), denoted $\mathsf{Succ}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})$ respectively, while the security of a primitive will be measured by the maximal successes or advantages of any adversary running within a time bounded by some t in the appropriate experiments, denoted $\mathsf{Succ}^{\mathsf{sec}}(t)$ and $\mathsf{Adv}^{\mathsf{sec}}(t)$ respectively. Adversaries can keep state during the different phases. We denote $\overset{\$}{\leftarrow}$ the outcome of a probabilistic algorithm or the sampling from a uniform distribution. See Appendix A.1 for more details.

2.1 Non-Interactive Labeled Commitments

A non-interactive labeled commitment scheme $\mathcal{C}$ is defined by three algorithms:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the CRS ρ to all other algorithms;
- $\mathsf{Com}^{\ell}(x)$ takes as input a label ℓ and a message x, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data (a.k.a. decommitment information). This is a probabilistic algorithm;
- $\mathsf{VerCom}^{\ell}(C, x, \delta)$ takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \perp$.

Using the experiments $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{hid}}(\mathfrak{K})$ and $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{bind}}(\mathfrak{K})$ defined in Figure 1, one can state the basic properties:

- *Correctness*: for all correctly generated CRS ρ, all commitments and opening data honestly generated pass the verification VerCom test: for all ℓ, x, if $(C, \delta) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(x)$, then $\mathsf{VerCom}^{\ell}(C, x, \delta) = 1$;
- *Hiding Property*: the commitment does not leak any information about the committed value. $\mathcal{C}$ is said (t, ε)-hiding if $\mathsf{Adv}_{\mathcal{C}}^{\mathtt{hid}}(t) \leq \varepsilon$.
- *Binding Property*: no adversary can open a commitment in two different ways. $\mathcal{C}$ is said (t, ε)-binding if $\mathsf{Succ}_{\mathcal{C}}^{\mathtt{bind}}(t) \leq \varepsilon$.

Correctness is always perfectly required, and one can also require either the binding or the hiding property to be perfect.

The reader can remark that labels are actually useless in the hiding and the binding properties. But they will become useful in $\mathtt{E}^2$-commitment schemes introduced in the next section. This is somehow similar to encryption scheme: labels are useless with encryption schemes which are just IND-CPA, but are very useful with IND-CCA encryption schemes.

2.2 Perfectly Binding Commitments: Public-Key Encryption

To get perfectly binding commitments, classical instantiations are public-key encryption schemes, which additionally provide extractability (see below). The encryption algorithm is indeed the commitment algorithm, and the random coins become the opening data that allow to check the correct procedure of the commit phase. The hiding property relies on the indistinguishability (`IND-CPA`), which is computationally achieved, whereas the binding property relies on the correctness of the encryption scheme and is perfect.

Let us define the ElGamal-based commitment scheme:

– $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses a cyclic group $\mathbb{G}$ of prime order p, g a generator for this group and a random scalar $z \xleftarrow{\$} \mathbb{Z}_p$. It sets the CRS $\rho = (\mathbb{G}, g, h = g^z)$;
– $\mathsf{Com}(M)$, for $M \in \mathbb{G}$, chooses a random element $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the pair $(C = (u = g^r, e = h^r \cdot M), \delta = r)$;
– $\mathsf{VerCom}(C = (u, e), M, \delta = r)$ checks whether $C = (u = g^r, e = h^r \cdot M)$.

This commitment scheme is hiding under the DDH assumption and perfectly binding. It is even extractable using the decryption key z: $M = e/u^z$. However, it is not labeled. The Cramer-Shoup encryption scheme [CS98] admits labels and is extractable and non-malleable, thanks to the `IND-CCA` security level. It is reviewed in Appendix B, with some extensions, since we will use it later in our applications.

2.3 Perfectly Hiding Commitments

The Pedersen scheme [Ped92] is the most famous perfectly hiding commitment: $\mathsf{Com}(m) = g^m h^r$ for a random scalar $r \xleftarrow{\$} \mathbb{Z}_p$ and a fixed basis $h \in \mathbb{G}$. The binding property relies on the DL assumption. Unfortunately, the opening value is the scalar r, which makes it hard to encrypt/decrypt efficiently, as required in our construction below. Haralambiev [Har11, Section 4.1.4] recently proposed a new commitment scheme, called TC4 (without label), with a group element as opening value:

– $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses an asymmetric pairing-friendly setting $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$, with an additional independent generator $T \in \mathbb{G}_2$. It sets the CRS $\rho = (\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, T, \mathbb{G}_T, p, e)$;
– $\mathsf{Com}(x)$, for $x \in \mathbb{Z}_p$, chooses a random element $r \xleftarrow{\$} \mathbb{Z}_p$ and outputs the pair $(C = g_2^r T^x, \delta = g_1^r)$;
– $\mathsf{VerCom}(C, x, \delta)$ checks whether $e(g_1, C/T^x) = e(\delta, g_2)$.

This commitment scheme is clearly perfectly hiding, since the groups are cyclic, and for any $C \in \mathbb{G}_2$, $x \in \mathbb{Z}_p$, there exists $\delta \in \mathbb{G}_1$ that satisfies $e(g_1, C/T^x) = e(\delta, g_2)$. More precisely, if $C = g_2^u$ and $T = g_2^t$, then $\delta = g_1^{u-tx}$ opens C to any x. The binding property holds under the DDH assumption in $\mathbb{G}_2$, as proven in [Har11, Section 4.1.4].

2.4 Equivocable Commitments

An equivocable commitment scheme $\mathcal{C}$ extends on the previous definition, with $\mathsf{SetupCom}$, Com, VerCom, and a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and

– $\mathsf{SimCom}^{\ell}(\tau)$ that takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
– $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ that takes as input a commitment C, a label ℓ, a message x, and an equivocation key eqk for this commitment, and outputs an opening data δ for C and ℓ on x.

```
$\mathsf{Exp}^{\mathtt{sim\text{-}ind\text{-}}b}_{\mathcal{A}}(\mathfrak{K})$
    $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(\ell, x, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot)}(\rho)$
    $\textbf{if } b = 0 \textbf{ then } (C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$
    $\textbf{else } (C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$
    $\textbf{return } \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \delta)$
```

```
$\mathsf{Exp}^{\mathtt{bind\text{-}ext}}_{\mathcal{A}}(\mathfrak{K})$
    $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(C, \ell, x, \delta) \xleftarrow{\$} \mathcal{A}^{\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
    $x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
    $\textbf{if } x' = x \textbf{ then return } 0$
    $\textbf{else return } \mathsf{VerCom}^{\ell}(C, x, \delta)$
```

Fig. 2. Simulation Indistinguishability and Binding Extractability

Let us denote SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$. Three additional properties are then associated: a *correctness* property, and two *indistinguishability* properties, which all together imply the *hiding* property.

- *Trapdoor Correctness*: all simulated commitments can be opened on any message: for all ℓ, x, if $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$, then $\mathsf{VerCom}^{\ell}(C, x, \delta) = 1$;
- *Setup Indistinguishability*: one cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$. $\mathcal{C}$ is said (t, ε)-setup-indistinguishable if the two distributions for ρ are (t, ε)-computationally indistinguishable. We denote $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t)$ the distance between the two distributions.
- *Simulation Indistinguishability*: one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments. $\mathcal{C}$ is said (t, ε)-simulation-indistinguishable if $\mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t) \leq \varepsilon$ (see the experiments $\mathsf{Exp}^{\mathtt{sim\text{-}ind\text{-}}b}_{\mathcal{A}}(\mathfrak{K})$ in Figure 2).

More precisely, when the trapdoor correctness is satisfied, since commitments generated by SimCom are perfectly hiding (they can be opened in any way using $\mathsf{OpenCom}$), $\mathsf{Adv}^{\mathtt{hid}}_{\mathcal{C}}(t) \leq \mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t)$.

Definition 1 (Equivocable Commitment). *A commitment scheme $\mathcal{C}$ is said (t, ε)-equivocable if, first, the basic commitment scheme satisfies the correctness property and is both (t, ε)-binding and (t, ε)-hiding, and, secondly, the additional algorithms guarantee the trapdoor correctness and make it both (t, ε)-setup-indistinguishable and (t, ε)-simulation-indistinguishable.*
One denotes $\mathsf{Adv}^{\mathtt{equiv}}_{\mathcal{C}}(t)$ the maximum of $\mathsf{Succ}^{\mathtt{bind}}_{\mathcal{C}}(t)$, $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{C}}(t)$, and $\mathsf{Adv}^{\mathtt{sim\text{-}ind}}_{\mathcal{C}}(t)$; it should be upper-bounded by ε.

2.5 Extractable Commitments

An extractable commitment scheme $\mathcal{C}$ also extends on the initial definition, with $\mathsf{SetupCom}$, Com, VerCom, as well as the second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and

- $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

As above, three additional properties are then associated: a *correctness* property, and the *setup indistinguishability*, but also an *extractability* property, which implies, together with the setup indistinguishability, the *binding* property:

- *Trapdoor Correctness*: all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$;
- *Setup Indistinguishability*: as above;
- *Binding Extractability*: one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x. $\mathcal{C}$ is said (t, ε)-binding-extractable if $\mathsf{Succ}^{\mathtt{bind\text{-}ext}}_{\mathcal{C}}(t) \leq \varepsilon$ (see the experiment $\mathsf{Exp}^{\mathtt{bind\text{-}ext}}_{\mathcal{A}}(\mathfrak{K})$ in Figure 2).

More precisely, when one breaks the binding property with $(C, \ell, x_0, \delta_0, x_1, \delta_1)$, if the extraction oracle outputs $x' = x_0$, then one can output (C, ℓ, x_1, δ_1), otherwise one can output (C, ℓ, x_0, δ_0). In both cases, this breaks the binding-extractability: $\mathsf{Adv}_{\mathcal{C}}^{\texttt{bind}}(t) \leq \mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + \mathsf{Succ}_{\mathcal{C}}^{\texttt{bind-ext}}(t)$.

Definition 2 (Extractable Commitment). *A commitment scheme $\mathcal{C}$ is said (t, ε)-extractable if, first, the basic commitment scheme satisfies the correctness property and is both (t, ε)-binding and (t, ε)-hiding, and, secondly, the additional algorithms guarantee the trapdoor correctness and make it both (t, ε)-setup-indistinguishable and (t, ε)-binding-extractable.*
One denotes $\mathsf{Adv}_{\mathcal{C}}^{\texttt{ext}}(t)$ the maximum of $\mathsf{Adv}_{\mathcal{C}}^{\texttt{hid}}(t)$, $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t)$, and $\mathsf{Succ}_{\mathcal{C}}^{\texttt{bind-ext}}(t)$; it should be upper-bounded by ε.

3 Equivocable and Extractable Commitments

3.1 $\mathtt{E}^2$-Commitments: Equivocable and Extractable

Public-key encryption schemes are perfectly binding commitments that are additionally extractable. The Pedersen and Haralambiev commitments are perfectly hiding commitments that are additionally equivocable. But none of them have the two properties at the same time. This is now our goal.

Definition 3 ($\mathtt{E}^2$-Commitment). *A commitment scheme $\mathcal{C}$ is said (t, ε)-$\mathtt{E}^2$(equivocable and extractable) if the indistinguishable setup algorithm outputs a common trapdoor that allows both equivocability and extractability. If one denotes $\mathsf{Adv}_{\mathcal{C}}^{\texttt{e}^2}(t)$ the maximum of the values $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t)$, $\mathsf{Adv}_{\mathcal{C}}^{\texttt{sim-ind}}(t)$, and $\mathsf{Succ}_{\mathcal{C}}^{\texttt{bind-ext}}(t)$, then it should be upper-bounded by ε.*

But with such a common trapdoor, the adversary could exploit the equivocation queries to break extractability and extraction queries to break equivocability. Stronger notions can thus be defined, using the experiments $\mathsf{Exp}_{\mathcal{A}}^{\texttt{s-sim-ind-}b}(\mathfrak{K})$ and $\mathsf{Exp}_{\mathcal{A}}^{\texttt{s-bind-ext}}(\mathfrak{K})$ in Figure 3, in which SCom is supposed to store each query/answer (ℓ, x, C) in a list Λ and ExtCom-queries on such an SCom-output (ℓ, C) are answered by x (as it would be when using Com instead of SCom).

- *Strong Simulation Indistinguishability*: one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom). $\mathcal{C}$ is said (t, ε)-strongly-simulation-indistinguishable if $\mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) \leq \varepsilon$;
- *Strong Binding Extractability* (informally introduced in [CLOS02] as "simulation extractability"): one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data (not given by SCom) to an input x while the commitment does not extract to x, even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom). $\mathcal{C}$ is said (t, ε)-strongly-binding-extractable if $\mathsf{Succ}_{\mathcal{C}}^{\texttt{s-bind-ext}}(t) \leq \varepsilon$.

They both imply the respective weaker notions since they just differ by giving access to the ExtCom-oracle in the former game, and to the SCom oracle in the latter. We insist that ExtCom-queries on SCom-outputs are answered by the related SCom-inputs. Otherwise, the former game would be void. In addition, VerCom always rejects inputs with $x = \bot$, which is useful in the latter game.

3.2 UC-Secure Commitments

The security definition for commitment schemes in the UC framework was presented by Canetti and Fischlin [CF01], refined by Canetti [Can05]. The ideal functionality is presented in Figure 4, where a *public delayed output* is an output first sent to the adversary $\mathcal{S}$ that eventually decides if and when the message is actually delivered to the recipient. In case of corruption of the committer,

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{s-sim-ind-}b}(\mathfrak{K})$
    $(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$;
    $(\ell, x, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
    if $b = 0$ then $(C,\delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$
    else $(C,\delta) \xleftarrow{\$} \mathsf{SCom}^{\ell}(\tau, x)$
    return $\mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \delta)$
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{s-bind-ext}}(\mathfrak{K})$
    $(\rho,\tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(C, \ell, x, \delta) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
    $x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
    if $(\ell, x', C) \in \Lambda$ then return $0$
    if $x' = x$ then return $0$
    else return $\mathsf{VerCom}^{\ell}(C, x, \delta)$
```

Fig. 3. Strong Simulation Indistinguishability and Strong Binding Extractability

The functionality $\mathcal{F}_{\text{com}}$ is parametrized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

Commit phase: Upon receiving a query $(\texttt{Commit}, \text{sid}, \text{ssid}, P_i, P_j, x)$ from party P_i: record the tuple $(\text{sid}, \text{ssid}, P_i, P_j, x)$ and generate a *public delayed output* $(\texttt{Receipt}, \text{sid}, \text{ssid}, P_i, P_j)$ to P_j. Ignore further `Commit`-message with the same ssid from P_i.

Decommit phase. Upon receiving a query $(\texttt{Reveal}, \text{sid}, \text{ssid}, P_i, P_j)$ from party P_i: ignore the message if $(\text{sid}, \text{ssid}, P_i, P_j, x)$ is not recorded; otherwise mark the record $(\text{sid}, \text{ssid}, P_i, P_j)$ as revealed and generate a *public delayed output* $(\texttt{Revealed}, \text{sid}, \text{ssid}, P_i, P_j, x)$ to P_j. Ignore further `Reveal`-message with the same ssid from P_i.

Fig. 4. Ideal Functionality for Commitment Scheme $\mathcal{F}_{\text{com}}$

if this is before the `Receipt`-message for the receiver, the adversary chooses the committed value, otherwise it is provided by the ideal functionality, according to the `Commit`-message. Note this is actually the multiple-commitment functionality that allows multiple executions of the commitment protocol (multiple ssid's) for the same functionality instance (one sid). This avoids the use of joint-state UC [CR03].

Theorem 4. *A labeled E^2-commitment scheme $\mathcal{C}$, that is in addition strongly-binding-extractable or strongly-simulation-indistinguishable, is a non-interactive UC-secure commitment scheme in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels. More precisely, for any environment, its advantage in distinguishing the ideal world (with the ideal functionality from Figure 4) and the real world (with the commitment scheme $\mathcal{C}$) is bounded by both $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q_s \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{sim-ind}}(t) + \mathsf{Succ}_{\mathcal{C}}^{\texttt{s-bind-ext}}(t)$ and $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q_s \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) + \mathsf{Succ}_{\mathcal{C}}^{\texttt{bind-ext}}(t)$, where q_s is the number of concurrent sessions and t its running time.*

Proof. The full proof with the cost of the reduction are given in Appendix D, but let us provide here the simulator:

- when receiving a commitment C from the adversary, and thus either freshly generated by the adversary or a replay of a commitment C generated by the simulator in another session (with a different label), the simulator extracts the committed value x, and uses it to send a `Commit` message to the ideal functionality. A dummy value is used in case of bad extraction;
- when receiving a `Receipt`-message, which means that an honest player has committed a value, the simulator generates $(C, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\text{sid}, \text{ssid}, P_i, P_j)$, to send C during the commit phase of the honest player;
- when receiving (x, δ), if the verification succeeds, the simulator asks for a `Reveal` query to the ideal functionality;
- when receiving a `Revealed`-message on x, it then generates $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ to actually open the commitment.

Any corruption just reveals x earlier, which allows a correct simulation of the opening. $\square$

4 A Construction of Labeled $\mathtt{E}^2$-Commitment Scheme

4.1 Labeled Cramer-Shoup Encryption on Vectors

For our construction we use a variant of the Cramer-Shoup encryption scheme for vectors of messages. Let $\mathbb{G}$ be a cyclic group of order p, with two independent generators g and h. The secret decryption key is a random vector $\mathsf{sk} = (x_1, x_2, y_1, y_2, z) \overset{\$}{\leftarrow} \mathbb{Z}_p^5$ and the public encryption key is $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f = g^z, H)$, where H is randomly chosen in a collision-resistant hash function family $\mathcal{H}$ (actually, second-preimage resistance is enough). For a message-vector $\boldsymbol{M} = (M_i)_{i=1,\ldots,m} \in \mathbb{G}^m$, the multi-Cramer-Shoup encryption is defined as $m\text{-}\mathsf{MCS}^{\ell}_{\mathsf{pk}}(\boldsymbol{M}; (r_i)_i) = (\mathsf{CS}^{\ell}_{\mathsf{pk}}(M_i, \theta; r_i) = (u_i = g^{r_i}, v_i = h^{r_i}, e_i = f^{r_i} \cdot M_i, w_i = (cd^{\theta})^{r_i}))_i$, where $\theta = H(\ell, (u_i, v_i, e_i)_i)$ is the same for all the w_i's to ensure non-malleability contrary to what we would have if we had just concatenated Cramer-Shoup ciphertexts of the M_i's. Such a ciphertext $C = (u_i, v_i, e_i, w_i)_i$ is decrypted by $M_i = e_i/u_i^z$, after having checked the validity of the ciphertext, $w_i \overset{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$, for $i = 1, \ldots, m$. This multi-Cramer-Shoup encryption scheme, denoted MCS, is $\mathtt{IND\text{-}CCA}$ under the DDH assumption. It even verifies a stronger property $\mathtt{VIND\text{-}PO\text{-}CCA}$ (for Vector-Indistinguishability with Partial Opening under Chosen-Ciphertext Attacks), where the random coins r_i, of the common coordinates in the two challenge vectors, are published with the challenge ciphertext (partial-opening of the random coins, see Appendix B for more details). This will be useful for the security proof of our commitment $\mathcal{E}^2\mathcal{C}$ (see below):

$$\mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathsf{MCS}}(m, q_d, \gamma, t) \leq 2\gamma \cdot \mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t) + \frac{m(\gamma+1)q_d}{p}, \tag{1}$$

where m is the length of the encrypted vectors, q_d is the maximal number of decryption queries, and γ the bound on the number of distinct components in the two challenge vectors; and where $\mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}$ and $\mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t)$ are respectively the maximum advantage of an adversary against the DDH problem and the maximum probability for an adversary to find a collision for $H \overset{\$}{\leftarrow} \mathcal{H}$, in time t.

4.2 Construction

In this section, we provide a concrete construction $\mathcal{E}^2\mathcal{C}$, inspired from [CF01,CLOS02,ACP09], with the above multi-Cramer-Shoup encryption (as an extractable commitment scheme) and the TC4 Haralambiev's equivocable commitment scheme [Har11, Section 4.1.4]. The latter will allow equivocability while the former will provide extractability:

- $\mathsf{SetupComT}(1^{\mathfrak{K}})$ generates a pairing-friendly setting $(\mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, p, e)$, with another independent generator h_1 of $\mathbb{G}_1$. It then generates the parameters of a Cramer-Shoup-based commitment in $\mathbb{G}_1$: $x_1, x_2, y_1, y_2, z \overset{\$}{\leftarrow} \mathbb{Z}_p$ and $H \overset{\$}{\leftarrow} \mathcal{H}$, and sets $\mathsf{pk} = (g_1, h_1, c = g_1^{x_1}h_1^{x_2}, d = g_1^{y_1}h_1^{y_2}, f_1 = g_1^z, H)$. It then chooses a random scalar $t \overset{\$}{\leftarrow} \mathbb{Z}_p$, and sets $T = g_2^t$. The CRS ρ is set as (pk, T) and the trapdoor τ is the decryption key (x_1, x_2, y_1, y_2, z) (a.k.a. extraction trapdoor) together with t (a.k.a. equivocation trapdoor). For $\mathsf{SetupCom}(1^{\mathfrak{K}})$, the CRS is generated the same way, but forgetting the scalars, and thus without any trapdoor;
- $\mathsf{Com}^{\ell}(\boldsymbol{M})$, for $\boldsymbol{M} = (M_i)_i \in \{0,1\}^m$ and a label ℓ, works as follows:
  - For $i = 1, \ldots, m$, it chooses a random scalar $r_{i,M_i} \overset{\$}{\leftarrow} \mathbb{Z}_p$, sets $r_{i,1-M_i} = 0$, and commits to M_i, using the TC4 commitment scheme with r_{i,M_i} as randomness: $a_i = g_2^{r_{i,M_i}} T^{M_i}$, and sets $d_{i,j} = g_1^{r_{i,j}}$ for $j = 0, 1$, which makes d_{i,M_i} the opening value for a_i to M_i; Let us also write $\boldsymbol{a} = (a_1, \ldots, a_m)$, the tuple of commitments.
  - For $i = 1, \ldots, m$ and $j = 0, 1$, it gets $\boldsymbol{b} = (b_{i,j})_{i,j} = 2m\text{-}\mathsf{MCS}^{\ell'}_{\mathsf{pk}}(\boldsymbol{d}; \boldsymbol{s})$, that is $(u_{i,j}, v_{i,j}, e_{i,j}, w_{i,j})_{i,j}$, where $\boldsymbol{d} = (d_{i,j})_{i,j}$ computed above, $\boldsymbol{s} = (s_{i,j})_{i,j} \overset{\$}{\leftarrow} \mathbb{Z}_p^{2m}$, and $\ell' = (\ell, \boldsymbol{a})$.

  The commitment is $C = (\boldsymbol{a}, \boldsymbol{b})$, and the opening information is the m-tuple $\delta = (s_{1,M_1}, \ldots, s_{m,M_m})$.

– $\mathsf{VerCom}^{\ell}(C, \boldsymbol{M}, \delta)$ checks the validity of the ciphertexts b_{i,M_i} with s_{i,M_i} and θ computed on the full ciphertext C, extracts d_{i,M_i} from b_{i,M_i} and s_{i,M_i}, and checks whether $e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2)$, for $i = 1, \dots, m$.
– $\mathsf{SimCom}^{\ell}(\tau)$ takes as input the equivocation trapdoor, namely t, and outputs $C = (\boldsymbol{a}, \boldsymbol{b})$ and $\mathsf{eqk} = \boldsymbol{s}$, where
  - For $i = 1, \dots, m$, it chooses a random scalar $r_{i,0} \xleftarrow{\$} \mathbb{Z}_p$, sets $r_{i,1} = r_{i,0} - t$, and commits to both 0 and 1, using the TC4 commitment scheme with $r_{i,0}$ and $r_{i,1}$ as respective randomness: $a_i = g_2^{r_{i,0}} = g_2^{r_{i,1}} T$, and $d_{i,j} = g_1^{r_{i,j}}$ for $j = 0, 1$, which makes $d_{i,j}$ the opening value for a_i to the value $j \in \{0, 1\}$. This leads to $\boldsymbol{a}$;
  - $\boldsymbol{b}$ is built as above: $\boldsymbol{b} = (b_{i,j})_{i,j} = 2m\text{-}\mathsf{MCS}^{\ell'}_{\mathsf{pk}}(\boldsymbol{d}; \boldsymbol{s})$, with random scalars $(s_{i,j})_{i,j}$.
– $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})$ simply extracts the useful values from $\mathsf{eqk} = \boldsymbol{s}$ to make the opening value $\delta = (s_{1,M_1}, \dots, s_{m,M_m})$ in order to open to $\boldsymbol{M} = (M_i)_i$.
– $\mathsf{ExtCom}^{\ell}(\tau, C)$ takes as input the extraction trapdoor, namely the Cramer-Shoup decryption key. Given $\boldsymbol{b}$, it can decrypt all the $b_{i,j}$ into $d_{i,j}$ and check whether $e(g_1, a_i/T^j) = e(d_{i,j}, g_2)$ or not. If, for each i, exactly one $j = M_i$ satisfies the equality, then the extraction algorithm outputs $(M_i)_i$, otherwise (no correct decryption or ambiguity with several possibilities) it outputs $\perp$.

4.3 Security Properties

The above commitment scheme $\mathcal{E}^2\mathcal{C}$ is a labeled $\mathtt{E}^2$-commitment, with both strong-simulation-indistinguishability and strong-binding-extractability, under the DDH assumptions in both $\mathbb{G}_1$ and $\mathbb{G}_2$. It is thus a UC-secure commitment scheme. The stronger $\mathtt{VIND\text{-}PO\text{-}CCA}$ security notion for the encryption scheme is required because the $\mathsf{SCom}/\mathsf{Com}$ oracle does not only output the commitment (and thus the ciphertexts) but also the opening values which include the random coins of the encryption, but just for the plaintext components that are the same in the two vectors, since the two vectors only differ for unnecessary data (namely the $d_{i,1-M_i}$'s) in the security proof. More details can be found in the full proof, in Appendix D, which leads to $\mathsf{Adv}^{\mathtt{setup\text{-}ind}}_{\mathcal{E}^2\mathcal{C}}(t) = 0$, $\mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{E}^2\mathcal{C}}(t) \leq \mathsf{Adv}^{\mathtt{vind\text{-}po\text{-}cca}}_{\mathsf{MCS}}(2m, q_d, m, t)$, and $\mathsf{Succ}^{\mathtt{s\text{-}bind\text{-}ext}}_{\mathcal{E}^2\mathcal{C}}(t) \leq q_c \cdot \mathsf{Adv}^{\mathtt{s\text{-}sim\text{-}ind}}_{\mathcal{E}^2\mathcal{C}}(t) + \mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}_2}(t)$, where q_c is the number of SCom-queries and q_d the number of ExtCom-queries.

5 SPHF-Friendly Commitments

5.1 Smooth Projective Hash Functions

Projective hash function families were first introduced by Cramer and Shoup [CS02], but we here use the definitions of Gennaro and Lindell [GL03], provided to build secure password-based authenticated key exchange protocols, together with non-malleable commitments.

Let X be the domain of these functions and let L be a certain subset of this domain (a language). A key property of these functions is that, for words C in L, their values can be computed by using either a *secret* hashing key hk or a *public* projection key hp but with a witness w of the fact that C is indeed in L:

– $\mathsf{HashKG}(L)$ generates a hashing key hk for the language L;
– $\mathsf{ProjKG}(\mathsf{hk}, L, C)$ derives the projection key hp, possibly depending on the word C;
– $\mathsf{Hash}(\mathsf{hk}, L, C)$ outputs the hash value from the hashing key, on any word $C \in X$;
– $\mathsf{ProjHash}(\mathsf{hp}, L, C, w)$ outputs the hash value from the projection key hp, and the witness w, for $C \in L$.

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{robust}}(\mathfrak{K})$
    $(\rho,\tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(C,\ell) \overset{\$}{\leftarrow} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$
    $x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
    $\textbf{if } (\ell, x', C) \in \Lambda \textbf{ then return } 0$
    $\textbf{if } \exists x \neq x', \exists \delta, \ \mathsf{VerCom}^{\ell}(C, x, \delta) \textbf{ then return } 1$
    $\textbf{else return } 0$
```

Fig. 5. Robustness

The set of hash values is called the *range* of the SPHF and is denoted Π. The *correctness* of the SPHF assures that if $C \in L$ with w a witness of this fact, then $\mathsf{Hash}(\mathsf{hk}, L, C) = \mathsf{ProjHash}(\mathsf{hp}, L, C, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $C \notin L$, $\mathsf{Hash}(\mathsf{hk}, L, C)$ is *statistically* indistinguishable from a random element, even knowing hp.

Note that HashKG and ProjKG can just depend partially on L (a superset L') and not at all on C: we then note $\mathsf{HashKG}(L')$ and $\mathsf{ProjKG}(\mathsf{hk}, L', \perp)$ (see [BBC+13] for more details on GL-SPHF and KV-SPHF and language definitions).

5.2 Robust Commitments

For a long time, SPHFs have been used to implicitly check some statements, on language membership, such as "C indeed encrypts x". This easily extends to perfectly binding commitments with labels: $L_x = \{(\ell, C) | \exists \delta, \mathsf{VerCom}^{\ell}(C, x, \delta) = 1\}$. But when commitments are equivocable, this intuitively means that a commitment C with the label ℓ contains any x and is thus in all the languages L_x. In order to be able to use SPHFs with $\mathtt{E}^2$-commitments, we want the commitments generated by polynomially-bounded adversaries to be perfectly binding, and thus to belong to at most one language L_x. We thus need a *robust verification* property for such $\mathtt{E}^2$*-commitments.*

Definition 5 (Robustness). *One cannot produce a commitment and a label that extracts to x' (possibly $x' = \perp$) such that there exists a valid opening data to a different input x, even with oracle access to the extraction oracle (*ExtCom*) and to fake commitments (using* SCom*). $\mathcal{C}$ is said (t, ε)-robust if $\mathsf{Succ}_{\mathcal{C}}^{robust}(t) \leq \varepsilon$, according to the experiment $\mathsf{Exp}_{\mathcal{A}}^{robust}(\mathfrak{K})$ in Figure 5.*

It is important to note that the latter experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{robust}}(\mathfrak{K})$ may not be run in polynomial time. Robustness implies strong-binding-extractability.

5.3 Properties of SPHF-Friendly Commitments

We are now ready to define SPHF-friendly commitments, which admit an SPHF on the languages $L_x = \{(\ell, C) | \exists \delta, \mathsf{VerCom}^{\ell}(C, x, \delta) = 1\}$, and to discuss about them:

Definition 6 (SPHF-Friendly Commitments). *An SPHF-friendly commitment is an $\mathtt{E}^2$-commitment that admits an SPHF on the languages L_x, and that is both strongly-simulation-indistinguishable and robust.*

Let us consider such a family $\mathcal{F}$ of SPHFs on languages L_x for $x \in X$, with X a non trivial set (with at least two elements), with hash values in the set G. From the smoothness of the SPHF on L_x, one can derive the two following properties on SPHF-friendly commitments, modeled by the experiments in Figure 6. The first notion of *smoothness* deals with adversary-generated commitments, that are likely perfectly binding from the robustness, while the second notion of *pseudo-randomness* deals with simulated commitments, that are perfectly hiding. They are inspired by the security games from [GL03].

In both security games, note that when hk and hp do not depend on x nor on C, and when the smoothness holds even if the adversary can choose C after having seen hp (*i.e.*, the SPHF is actually a $\mathsf{KV\text{-}SPHF}$ [BBC+13]), they can be generated from the beginning of the games, with hp given to the adversary much earlier.

Smoothness of SPHF-Friendly Commitments. If the adversary $\mathcal{A}$, with access to the oracles SCom and ExtCom, outputs a fresh commitment (ℓ, C) that extracts to $x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$, then the robustness guarantees that for any $x \neq x'$, $(\ell, C) \notin L_x$ (excepted with small probability), and thus the distribution of the hash value is statistically indistinguishable from the random distribution, even when knowing hp. In the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{c\text{-}smooth}}(\mathfrak{K})$, we let the adversary choose x, and we have: $\mathsf{Adv}_{\mathcal{C},\mathcal{F}}^{\mathtt{c\text{-}smooth}}(t) \leq \mathsf{Succ}_{\mathcal{C}}^{\mathtt{robust}}(t) + \mathsf{Adv}_{\mathcal{F}}^{\mathtt{smooth}}$.

Pseudo-Randomness of SPHF on Robust Commitments. If the adversary $\mathcal{A}$ is given a commitment C by SimCom with label ℓ, adversary-chosen, even with access to the oracles SCom and ExtCom, then for any x, it cannot distinguish the hash value of (ℓ, C) on language L_x from a random value, even being given hp, since C could have been generated as $\mathsf{Com}^{\ell}(x'')$ for some $x'' \neq x$, which excludes it to belong to L_x, under the robustness. In the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{c\text{-}ps\text{-}rand}}(\mathfrak{K})$, we let the adversary choose (ℓ, x), and we have: $\mathsf{Adv}_{\mathcal{C},\mathcal{F}}^{\mathtt{c\text{-}ps\text{-}rand}}(t) \leq \mathsf{Adv}_{\mathcal{C}}^{\mathtt{s\text{-}sim\text{-}ind}}(t) + \mathsf{Succ}_{\mathcal{C}}^{\mathtt{robust}}(t) + \mathsf{Adv}_{\mathcal{F}}^{\mathtt{smooth}}$.

Strong Pseudo-Randomness. Besides these two properties which hold for any SPHF-friendly commitment, we need a third property for our one-round PAKE protocol. This property, called strong pseudo-randomness, is defined by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{c\text{-}s\text{-}ps\text{-}rand}}(\mathfrak{K})$ depicted in Figure 6. It is a strong version of the pseudo-randomness where the adversary is also given the hash value of a commitment of its choice (obviously not generated by SCom or SimCom though, hence the test with Λ which also contains (C, ℓ, x)). This property only makes sense when the projection key does not depend on the word C to be hashed. It thus applies to $\mathsf{KV\text{-}SPHF}$ only.

5.4 Our Commitment Scheme $\mathcal{E}^2\mathcal{C}$ is SPHF-Friendly

In order to be *SPHF-friendly*, the commitment first needs to be *strongly-simulation-indistinguishable* and *robust*. We have already shown the former property, and the latter is also proven in Appendix D. One additionally needs an SPHF able to check the verification equation: using the notations from Section 4.2, $C = (\boldsymbol{a}, \boldsymbol{b})$ is a commitment of $\boldsymbol{M} = (M_i)_i$, if there exist $\delta = (s_{1,M_1}, \ldots, s_{m,M_m})$ and $(d_{1,M_1}, \ldots, d_{m,M_m})$ such that $b_{i,M_i} = (u_{i,M_i}, v_{i,M_i}, e_{i,M_i}, w_{i,M_i}) = \mathsf{CS}_{\mathsf{pk}}^{\ell'}(d_{i,M_i}, \theta; s_{i,M_i})$ (with a particular θ) and $e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2)$, for $i = 1, \ldots, m$. Since e is non-degenerated, we can eliminate the need of d_{i,M_i}, by lifting everything in $\mathbb{G}_T$, and checking that, first, the ciphertexts are all valid:

$$e(u_{i,M_i}, g_2) = e(g_1^{s_{i,M_i}}, g_2) \qquad e(v_{i,M_i}, g_2) = e(h_1^{s_{i,M_i}}, g_2) \qquad e(w_{i,M_i}, g_2) = e((cd^{\theta})^{s_{i,M_i}}, g_2)$$

and, second, the plaintexts satisfy the appropriate relations:

$$e(e_{i,M_i}, g_2) = e(f_1^{s_{i,M_i}}, g_2) \cdot e(g_1, a_i/T^{M_i}).$$

From these expressions we derive several constructions of such SPHFs in Appendix C, and focus here on the most interesting ones for the following applications:

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-smooth-}b}(\mathfrak{K})$
    $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(C, \ell, x, \mathsf{state}) \overset{\$}{\leftarrow} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho);\ x' \leftarrow \mathsf{ExtCom}^{\ell}(\tau, C)$
    $\textbf{if } (\ell, x', C) \in \Lambda \textbf{ then return } 0$
    $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(L_x);\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L_x, (\ell, C))$
    $\textbf{if } b = 0 \lor x' = x \textbf{ then } H \leftarrow \mathsf{Hash}(\mathsf{hk}, L_x, (\ell, C))$
    $\textbf{else } H \overset{\$}{\leftarrow} \Pi$
    $\textbf{return } \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, \mathsf{hp}, H)$
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-ps-rand-}b}(\mathfrak{K})$
    $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(\ell, x, \mathsf{state}) \overset{\$}{\leftarrow} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho);\ C \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$
    $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(L_x); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L_x, (\ell, C))$
    $\textbf{if } b = 0 \textbf{ then } H \leftarrow \mathsf{Hash}(\mathsf{hk}, L_x, (\ell, C))$
    $\textbf{else } H \overset{\$}{\leftarrow} \Pi$
    $\textbf{return } \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \mathsf{hp}, H)$
```

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-}b}(\mathfrak{K})$
    $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(\ell, x, \mathsf{state}) \overset{\$}{\leftarrow} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho);\ C \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$
    $\mathsf{hk} \overset{\$}{\leftarrow} \mathsf{HashKG}(L_x); \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L_x, \perp)$
    $\textbf{if } b = 0 \textbf{ then } H \leftarrow \mathsf{Hash}(\mathsf{hk}, L_x, (\ell, C))$
    $\textbf{else } H \overset{\$}{\leftarrow} \Pi$
    $(\ell', C', \mathsf{state}) \overset{\$}{\leftarrow} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \mathsf{hp}, H)$
    $\textbf{if } (\ell', ?, C') \in \Lambda \textbf{ then } H' \leftarrow \perp$
    $\textbf{else } H' \leftarrow \mathsf{Hash}(\mathsf{hk}, L_x, (\ell', C')$
    $\textbf{return } \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(H')$
```

Fig. 6. Smoothness, Pseudo-Randomness and Strong Pseudo-Randomness

Table 1. Comparison with existing non-interactive UC-secure commitments with a single global CRS (m = bit-length of the committed value, $\mathfrak{K}$ = security parameter)

	SPHF-Friendly	Commitment C	Decommitment δ	Assumption
[ACP09][a]	yes	$(m + 16m\mathfrak{K}) \times \mathbb{G}$	$2m\mathfrak{K} \times \mathbb{Z}_p$	DDH
[FLM11], 1	no	$5 \times \mathbb{G}$	$16 \times \mathbb{G}$	DLIN
[FLM11], 2	no	$37 \times \mathbb{G}$	$3 \times \mathbb{G}$	DLIN
this paper	yes	$8m \times \mathbb{G}_1 + m \times \mathbb{G}_2$	$m \times \mathbb{Z}_p$	SXDH

[a] slight variant without one-time signature but using labels for the IND-CCA security of the multi-Cramer-Shoup ciphertexts, as in our new scheme, and supposing that an element in the cyclic group $\mathbb{G}$ has size $2\mathfrak{K}$, to withstand generic attacks.

– First, when C is sent in advance (known when generating hp), as in the OT protocol described in Section 7, for $\mathsf{hk} = (\eta, \alpha, \beta, \mu, \varepsilon) \xleftarrow{\$} \mathbb{Z}_p^5$, and $\mathsf{hp} = (\varepsilon, \mathsf{hp}_1 = g_1^\eta h_1^\alpha f_1^\beta (cd^\theta)^\mu) \in \mathbb{Z}_p \times \mathbb{G}_1$:

$$\begin{aligned} H &= \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \\ &\overset{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^\eta \cdot v_{i,M_i}^\alpha, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^\beta \cdot e(w_{i,M_i}^\mu, g_2) \right)^{\varepsilon^{i-1}} \\ &= e(\textstyle\prod_i \mathsf{hp}_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \overset{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, \delta) = H'. \end{aligned}$$

– Then, when C is not necessarily known for computing hp, as in the one-round PAKE, described in Section 6, for $\mathsf{hk} = (\eta_{i,1}, \eta_{i,2}, \alpha_i, \beta_i, \mu_i)_i \xleftarrow{\$} \mathbb{Z}_p^{5m}$, and $\mathsf{hp} = (\mathsf{hp}_{i,1} = g_1^{\eta_{i,1}} h_1^{\alpha_i} f_1^{\beta_i} c^{\mu_i}, \mathsf{hp}_{i,2} = g_1^{\eta_{i,2}} d^{\mu_i})_i \in \mathbb{G}_1^{2m}$:

$$\begin{aligned} H &= \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \\ &\overset{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^{(\eta_{i,1}+\theta\eta_{i,2})} \cdot v_{i,M_i}^{\alpha_i}, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\beta_i} \cdot e(w_{i,M_i}^{\mu_i}, g_2) \right) \\ &= e(\textstyle\prod_i (\mathsf{hp}_{i,1}\mathsf{hp}_{i,2}^\theta)^{s_{i,M_i}}, g_2) \overset{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, \delta) = H'. \end{aligned}$$

This SPHF verifies the strong pseudo-randomness property, as shown in Appendix D.2. More precisely, we have: $\mathsf{Succ}_{\mathcal{E}^2\mathcal{C},\mathcal{F}}^{\texttt{c-s-ps-rand}}(t) \leq 2 \cdot \mathsf{Succ}_{\mathcal{E}^2\mathcal{C}}^{\texttt{robust}}(t) + 2 \cdot \mathsf{Succ}_{\mathcal{E}^2\mathcal{C}}^{\texttt{s-bind-ext}}(t) + 2 \cdot \mathsf{Adv}_{\mathcal{F}}^{\texttt{smooth}}$.

5.5 Complexity and Comparisons

As summarized in Table 1, the communication complexity is linear in $m \cdot \mathfrak{K}$ (where m is the bit-length of the committed value and $\mathfrak{K}$ is the security parameter), which is much better than [ACP09] that was linear in $m \cdot \mathfrak{K}^2$, but asymptotically worse than the two proposals in [FLM11] that are linear in $\mathfrak{K}$, and thus independent of m (as long as $m = \mathcal{O}(\mathfrak{K})$).

Basically, the first scheme in [FLM11] consists of a Cramer-Shoup-like encryption C of the message x, and a perfectly-sound Groth-Sahai [GS08] NIZK π that C contains x. The actual commitment is C and the opening value on x is $\delta = \pi$. The trapdoor-setup provides the Cramer-Shoup decryption key and changes the Groth-Sahai setup to the perfectly-hiding setting. The indistinguishable setups of the Groth-Sahai mixed commitments ensure the setup-indistinguishability. The extraction algorithm uses the Cramer-Shoup decryption algorithm, while the equivocation uses the simulator of the NIZK. The `IND-CCA` security notion for C and the computational soundness of π make it strongly-binding-extractable, the `IND-CCA` security notion and the zero-knowledge property of the NIZK provide the strong-simulation-indistinguishability. It is thus UC-secure. However, the verification is not robust: because of the perfectly-hiding setting of Groth-Sahai proofs, for any ciphertext C and for any message x, there exists a proof π that makes the verification of C on x. As a consequence, it is not SPHF-friendly. The second construction is in the same vein: they cannot be used in the following applications.

Table 2. Comparison with existing UC-secure PAKE schemes

	Adaptive	One-round	Communication complexity	Assumption
[ACP09][a]	yes	no	$2 \times (2m + 22m\mathfrak{K}) \times \mathbb{G} + \mathrm{OTS}$[b]	DDH
[KV11]	no	yes	$\approx 2 \times 70 \times \mathbb{G}$	DLIN
[BBC+13]	no	yes	$2 \times 6 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH
this paper	yes	yes	$2 \times 10m \times \mathbb{G}_1 + 2 \times m \times \mathbb{G}_2$	SXDH

[a] with the commitment variant of note "a" of Table 1.
[b] OTS: one-time signature (public key size and signature size) to link the flows in the PAKE protocol.

6 Password-Authenticated Key Exchange

6.1 A Generic Construction

The ideal functionality of a Password-Authenticated Key Exchange (PAKE) is depicted in Appendix A.3. It has been proposed in [CHK+05]. In Figure 7, we describe a one-round PAKE that is UC-secure against adaptive adversaries, assuming erasures. It can be built from any SPHF-friendly commitment scheme (that is E^2, strongly-simulation-indistinguishable, and robust as described in Section 5), if the SPHF is actually a KV-SPHF [BBC+13] and the algorithms HashKG and ProjKG do not need to know the committed value π (nor the word (ℓ, C) itself), for which the SPHF verifies the strong pseudo-randomness property. We thus denote L_π the language of the pairs (ℓ, C), where C is a commitment that opens to π under the label ℓ, and L the union of all the L_π (L does not depend on π). The proof of the following theorem, with the cost of the reduction, are given in Appendix D.

Theorem 7. *The Password-Authenticated Key-Exchange described on Figure 7 is UC-secure in the presence of adaptive adversaries, assuming erasures, as soon as the commitment scheme is SPHF-friendly with a KV-SPHF.*

6.2 Concrete Instantiation

Using our commitment $\mathcal{E}^2\mathcal{C}$ introduced Section 4 together with the second SPHF described Section 5 (which satisfies the above requirements for HashKG and ProjKG), one gets a quite efficient protocol, described in Appendix E. More precisely, for m-bit passwords, each player has to send $\mathsf{hp} \in \mathbb{G}_1^{2m}$ and $C \in \mathbb{G}_1^{8m} \times \mathbb{G}_2^{m}$, which means $10m$ elements from $\mathbb{G}_1$ and m elements from $\mathbb{G}_2$. In Table 2, we compare our new scheme with some previous UC-secure PAKE.

CRS: $\rho \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}})$.

Protocol execution by P_i with π_i:

1. P_i generates $\mathsf{hk}_i \overset{\$}{\leftarrow} \mathsf{HashKG}(L)$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, L, \perp)$
   and erases any random coins used for the generation
2. P_i computes $(C_i, \delta_i) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell_i}(\pi_i)$ with $\ell_i = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i stores δ_i, completely erases random coins used by Com
   and sends hp_i, C_i to P_j

Key computation: Upon receiving hp_j, C_j from P_j

1. P_i computes $H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, L_{\pi_i}, (\ell_i, C_i), \delta_i)$
   and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, L_{\pi_i}, (\ell_j, C_j))$ with $\ell_j = (\mathrm{sid}, \mathrm{ssid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = H'_i \cdot H_j$ and erases everything else, except π_i.

Fig. 7. UC-Secure PAKE from an SPHF-Friendly Commitment

7 Oblivious Transfer

7.1 A Generic Construction

The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Appendix A.3. It is inspired from [CKWZ13]. In Figure 8, we describe a 3-round OT that is UC-secure against adaptive adversaries, and a 2-round variant which is UC-secure against static adversaries. They can be built from any SPHF-friendly commitment scheme, where L_t is the language of the commitments that open to t under the associated label ℓ, and from any IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with plaintext size at least $\mathfrak{K}$, and from any Pseudo-Random Generator (PRG) F with input size equal to plaintext size, and output size equal to the size of the messages in the database. Details on encryption schemes and PRGs can be found in Appendix A.2. Notice the adaptive version can be seen as a variant of the static version where the last flow is sent over a somewhat secure channel, as in [CKWZ13]; and the preflow and pk and c are used to create this somewhat secure channel.

The proof of the following theorem, with the cost of the reduction, are given in Appendix D.

CRS: $\rho \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathtt{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$.

Pre-flow (for adaptive security only)**:**

1. P_i generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{param})$ for $\mathcal{E}$
2. P_i stores sk, completely erase random coins used by KeyGen, and sends pk to P_i

Index query on s**:**

1. P_j chooses a random value S, computes $R \leftarrow F(S)$ and encrypts S under pk: $c \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{pk}, S)$ (for adaptive security only; for static security: $c = \perp, R = 0$)
2. P_j computes $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(s)$ with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$
3. P_j stores δ, completely erases S and random coins used by Com and Encrypt, and sends C and c to P_i

Database input $(m_1, \ldots, m_k)$**:**

1. P_i decrypts $S \leftarrow \mathsf{Decrypt}(\mathsf{sk}, c)$ and gets $R \leftarrow F(S)$ (for static security: $R = 0$)
2. P_i computes $\mathsf{hk}_t \xleftarrow{\$} \mathsf{HashKG}(L_t)$, $\mathsf{hp}_t \leftarrow \mathsf{ProjKG}(\mathsf{hk}_t, L_t, (\ell, C))$,
   $K_t \leftarrow \mathsf{Hash}(\mathsf{hk}_t, L_t, (\ell, C))$, and $M_t \leftarrow R \oplus K_t \oplus m_t$, for $t = 1, \ldots, k$
3. P_i erases everything except $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$ and sends $(\mathsf{hp}_t, M_t)_t$ to P_j

Data recovery:
Upon receiving $(\mathsf{hp}_t, M_t)_{t=1,\ldots,k}$, P_j computes $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, L_s, (\ell, C), \delta)$ and gets $m_s \leftarrow R \oplus K_s \oplus M_s$. Then P_j erases everything except m_s and s

Fig. 8. UC-Secure 1-out-of-k OT from an SPHF-Friendly Commitment (for Adaptive and Static Security)

Theorem 8. *The two Oblivous Transfer schemes described in Figure 8 are UC-secure in the presence of adaptive adversaries and static adversaries respectively, assuming reliable erasures and authenticated channels, as soon as the commitment scheme is SPHF-friendly.*

7.2 Concrete Instantiation and Comparison

Using our commitment $\mathcal{E}^2\mathcal{C}$ introduced Section 4 together with the first SPHF described Section 5, one gets the protocol described in Appendix E, where the number of bits of the commited value is $m = \lceil \log k \rceil$. For the statically secure version, the communication cost is, in addition to the database $\boldsymbol{m}$ that is sent in $\boldsymbol{M}$ in a masked way, 1 element of $\mathbb{Z}_p$ and k elements of $\mathbb{G}_1$ (for **hp**, by using the same scalar ε for all hp_t's) for the sender, while the receiver sends $\lceil \log k \rceil$ elements of $\mathbb{G}_2$ (for $\boldsymbol{a}$) and $\lceil 8 \log k \rceil$ elements of $\mathbb{G}_1$ (for $\boldsymbol{b}$), in only two rounds. In the particular case of $k = 2$, the scalar can be avoided since the message consists of 1 bit, so our construction just requires: 2 elements from $\mathbb{G}_1$ for the sender, and 1 from $\mathbb{G}_2$ and 8 from $\mathbb{G}_1$ for the receiver, in two rounds. For the same security level (static corruptions in the CRS, with erasures), the best known solution from [CKWZ13] required to send at least 23 group elements and 7 scalars, in 4 rounds. If adaptive security is required, our construction requires 3 additional elements in $\mathbb{G}_1$ and 1 additional round, which gives a total of 13 elements in $\mathbb{G}_1$, in 3 rounds. This is also more efficient then the best known solution from [CKWZ13], which requires 26 group elements and 7 scalars, in 4 rounds.

Acknowledgments

We thank Ralf Küsters for his comments on a preliminary version. This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet. The third author was funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

References

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, August 2009. (Pages 1, 3, 9, 14, and 15.)

BBC$^+$13. Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475. Springer, 2013. Full version available on the Cryptology ePrint Archive as reports 2013/034 and 2013/341. (Pages 1, 11, 12, 15, 24, 25, and 26.)

BCL$^+$05. Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 361–377. Springer, August 2005. (Page 1.)

Bea96. Donald Beaver. Adaptive zero knowledge and computational equivocation (extended abstract). In *28th Annual ACM Symposium on Theory of Computing*, pages 629–638. ACM Press, May 1996. (Page 2.)

BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992. (Page 1.)

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111. Springer, March 2012. (Page 1.)

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001. (Pages 1 and 2.)

Can05. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. Cryptology ePrint Archive, Report 2000/067, 2005. `http://eprint.iacr.org/`. (Page 7.)

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, August 2001. (Pages 1, 2, 3, 7, and 9.)

CHK$^+$05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005. (Pages 1, 15, and 20.)

CK02. Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer, April / May 2002. (Page 1.)

CKWZ13. Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global crs. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *Public Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 73–88. Springer, 2013. (Pages 1, 3, 16, 17, and 20.)

CLOS02. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th Annual ACM Symposium on Theory of Computing*, pages 494–503. ACM Press, May 2002. (Pages 1, 3, 7, and 9.)

CR03. Ran Canetti and Tal Rabin. Universal composition with joint state. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 265–281. Springer, August 2003. (Page 8.)

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998. (Pages 1 and 5.)

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, April / May 2002. (Pages 1 and 10.)

DDN00. Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. (Page 2.)

FLM11. Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology – ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 468–485. Springer, December 2011. (Pages 3 and 14.)

GL03. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`. (Pages 1, 10, and 11.)

GMW87. Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game, or a completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229. ACM Press, May 1987. (Page 1.)

GMW91. Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, 1991. (Page 1.)

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008. (Page 14.)

Har11. Kristiyan Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications*. PhD thesis, New York University, 2011. (Pages 3, 5, and 9.)

HK07. Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 111–129. Springer, August 2007. (Page 1.)

HMQ04. Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 58–76. Springer, February 2004. (Page 2.)

Kal05. Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 78–95. Springer, May 2005. (Page 1.)

Kat07. Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128. Springer, May 2007. (Page 2.)

KOY01. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, May 2001. (Page 1.)

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer, March 2011. (Pages 1 and 15.)

NP01. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457. ACM-SIAM, January 2001. (Page 1.)

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology – CRYPTO'91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, August 1992. (Pages 3 and 5.)

PVW08. Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, August 2008. (Page 1.)

Rab81. Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981. (Page 1.)

A Notations

We first recall the classical definitions on distances of distribution, and the notions of success and advantage. We then review the basic cryptographic tools, with the corresponding security notions.

A.1 Distances, Advantage and Success

Statistical Distance. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be two probability distributions over a finite set $\mathcal{S}$ and let X_0 and X_1 be two random variables with these two respective distributions. The statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is also the statistical distance between X_0 and X_1:

$$\mathtt{Dist}(\mathcal{D}_0, \mathcal{D}_1) = \mathtt{Dist}(X_0, X_1) = \sum_{x \in \mathcal{S}} |\Pr[X_0 = x] - \Pr[X_1 = x]| .$$

If the statistical distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is less than or equal to ε, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are ε-close or are ε-statistically indistinguishable. If the $\mathcal{D}_0$ and $\mathcal{D}_1$ are 0-close, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are perfectly indistinguishable.

Success/Advantage. When one considers an experiment $\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A}}(\mathfrak{K})$ in which an adversary $\mathcal{A}$ plays a security game SEC, we denote $\mathsf{Succ}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{sec}}_{\mathcal{A}}(\mathfrak{K}) = 1]$ the success probability of this adversary. We additionally denote $\mathsf{Succ}^{\mathsf{sec}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Succ}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})\}$, the maximal success any adversary running within time t can get.

When one considers a pair of experiments $\mathsf{Exp}^{\mathsf{sec}\text{-}b}_{\mathcal{A}}(\mathfrak{K})$, for $b = 0, 1$, in which an adversary $\mathcal{A}$ plays a security game SEC, we denote $\mathsf{Adv}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[\mathsf{Exp}^{\mathsf{sec}-0}_{\mathcal{A}}(\mathfrak{K}) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{sec}-1}_{\mathcal{A}}(\mathfrak{K}) = 1\right]$ the advantage of this adversary. We additionally denote $\mathsf{Adv}^{\mathsf{sec}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Adv}^{\mathsf{sec}}(\mathcal{A}, \mathfrak{K})\}$, the maximal advantage any adversary running within time t can get.

Computational Distance. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be two probability distributions over a finite set $\mathcal{S}$ and let X_0 and X_1 be two random variables with these two respective distributions. The computational distance between $\mathcal{D}_0$ and $\mathcal{D}_1$ is the best advantage an adversary can get in distinguishing X_0 from X_1: $\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(\mathcal{A}, \mathfrak{K}) = \Pr[\mathcal{A}(X_0) = 1] - \Pr[\mathcal{A}(X_1) = 1]$, and thus $\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(\mathcal{A}, \mathfrak{K})\}$. When $\mathsf{Adv}^{\mathcal{D}_0, \mathcal{D}_1}(t) \leq \varepsilon$, we say that $\mathcal{D}_0$ and $\mathcal{D}_1$ are (t, ε)-computationally indistinguishable.

We can note that for two distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ that are ε-close, for any t and ε, $\mathcal{D}_0$ and $\mathcal{D}_1$ are (t, ε)-computationally indistinguishable.

A.2 Formal Definitions of the Basic Primitives

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions H_k from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^{\mathfrak{K}}$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any adversary $\mathcal{A}$ on a random function $H \stackrel{\$}{\leftarrow} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(\mathcal{A}, \mathfrak{K}) = \Pr\left[H \stackrel{\$}{\leftarrow} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(H) : H(m_0) = H(m_1)\right].$$

Labeled Encryption Scheme. A labeled public-key encryption scheme $\mathcal{E}$ is defined by four algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters $\mathtt{param}$ of the scheme;
- $\mathsf{KeyGen}(\mathtt{param})$ generates a pair of keys, the public encryption key pk and the private decryption key sk;

- $\mathsf{Encrypt}^{\ell}(\mathsf{pk}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key pk, using the random coins r;
- $\mathsf{Decrypt}^{\ell}(\mathsf{sk}, c)$ outputs the plaintext m encrypted in c under the label ℓ, or $\perp$ for an invalid ciphertext.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{pk}, \mathsf{sk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}^{\ell}(\mathsf{sk}, \mathsf{Encrypt}^{\ell}(\mathsf{pk}, m; r)) = m.$$

- *Indistinguishability under chosen-ciphertext attacks*: this security notion IND-CCA can be formalized by the following experiments $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{ind\text{-}cca\text{-}}b}(\mathfrak{K})$, where the adversary $\mathcal{A}$ transfers some internal state state between the various calls FIND and GUESS, and makes use of the oracle $\mathsf{ODecrypt}$:
  - $\mathsf{ODecrypt}^{\ell}(c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key sk. The input queries (ℓ, c) are added to the list CTXT.

```
$\mathsf{Exp}_{\mathcal{A}}^{\mathtt{ind\text{-}cca\text{-}}b}(\mathfrak{K})$
    $\mathtt{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$
    $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\mathtt{param})$
    $(\ell^*, m_0, m_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathtt{FIND} : \mathsf{pk})$
    $c^* \leftarrow \mathsf{Encrypt}^{\ell^*}(\mathsf{pk}, m_b)$
    $b' \leftarrow \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathsf{state}, \mathtt{GUESS} : c^*)$
    if $(\ell^*, c^*) \in \mathtt{CTXT}$ then return $0$
    else return $b'$
```

  According to the previous section, these experiments implicitly define the two advantages $\mathsf{Adv}_{\mathcal{E}}^{\mathtt{ind\text{-}cca}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}_{\mathcal{E}}^{\mathtt{ind\text{-}cca}}(t)$. One sometimes uses $\mathsf{Adv}_{\mathcal{E}}^{\mathtt{ind\text{-}cca}}(q_d, t)$ to bound the number of decryption queries.

Pseudo-Random Generators (PRGs). A pseudo-random generator (PRG) is a function F so that for a randomly chosen seed $S \xleftarrow{\$} \{0,1\}^{\mathfrak{K}}$ outputs a random-looking value in $\{0,1\}^{\ell}$, for some $\ell > \mathfrak{K}$.

The quality of a PRG is measured by the computational distance $\mathsf{Adv}_{F}^{\mathtt{prg}}(t)$ between the distributions of the outputs on random inputs from random values in $\{0,1\}^{\ell}$.

A.3 Ideal Functionalities

UC-Secure Oblivious Transfer. The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Figure 9. It is inspired from [CKWZ13].

UC-Secure Password-Authenticated Key Exchange. We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 10). It was described in [CHK+05]. The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of wrong guess (the session is then marked as `interrupted`), the two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving an input $(\texttt{Send}, \mathbf{sid}, \mathbf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j}, (\boldsymbol{m_1}, \ldots, \boldsymbol{m_k}))$ from party $\boldsymbol{P_i}$**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\text{sid}, \text{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ and reveal $(\texttt{Send}, \text{sid}, \text{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further `Send`-message with the same ssid from P_i.
- **Upon receiving an input $(\texttt{Receive}, \mathbf{sid}, \mathbf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j}, \boldsymbol{s})$ from party $\boldsymbol{P_j}$**, with $s \in \{1, \ldots, k\}$: record the tuple $(\text{sid}, \text{ssid}, P_i, P_j, s)$, and reveal $(\texttt{Receive}, \text{sid}, \text{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further `Receive`-message with the same ssid from P_j.
- **Upon receiving a message $(\texttt{Sent}, \mathbf{sid}, \mathbf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j})$ from the adversary $\boldsymbol{\mathcal{S}}$**: ignore the message if $(\text{sid}, \text{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\text{sid}, \text{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\texttt{Sent}, \text{sid}, \text{ssid}, P_i, P_j)$ to P_i and ignore further `Sent`-message with the same ssid from the adversary.
- **Upon receiving a message $(\texttt{Received}, \mathbf{sid}, \mathbf{ssid}, \boldsymbol{P_i}, \boldsymbol{P_j})$ from the adversary $\boldsymbol{\mathcal{S}}$**: ignore the message if $(\text{sid}, \text{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\text{sid}, \text{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\texttt{Received}, \text{sid}, \text{ssid}, P_i, P_j, m_s)$ to P_j and ignore further `Received`-message with the same ssid from the adversary.

Fig. 9. Ideal Functionality for 1-out-of-k Oblivious Transfer $\boldsymbol{\mathcal{F}_{(1,k)\text{-OT}}}$

with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password, are related passwords, are used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player, after the `NewKey`-query, it additionally learns the session key.

B Cramer-Shoup Encryption on Vectors

B.1 The Computational Assumption

Definition 9 (Decisional Diffie-Hellman (DDH)). *The Decisional Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b, g^c) for unknown random $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple).*

We denote by $\mathsf{Adv}_{\mathbb{G}}^{\mathsf{ddh}}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple in the group $\mathbb{G}$ within time t.

B.2 Description of the Cramer Shoup Encryption on Vectors

Labeled Cramer-Shoup Encryption. The labeled Cramer-Shoup encryption scheme works in a cyclic group $\mathbb{G}$ of prime order p, with two independent generators g and h. For random scalars $x_1, x_2, y_1, y_2, z \xleftarrow{\$} \mathbb{Z}_p$, we set $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ to be the private decryption key and $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f = g^z, H)$ to be the public encryption key, where H is a random collision-resistant hash function from $\mathcal{H}$ (actually, second-preimage resistance is enough).

If $M \in G$, the Cramer-Shoup encryption is defined as $\mathsf{CS}^{\ell}_{\mathsf{pk}}(M; r) = (u = g^r, v = h^r, e = f^r \cdot M, w = (cd^{\theta})^r)$, where $\theta = H(\ell, u, v, e)$. Such a ciphertext $C = (u, v, e, w)$ is decrypted by $M = e/u^z$, after having checked the validity of the ciphertext: $w \stackrel{?}{=} u^{x_1+\theta y_1}v^{x_2+\theta y_2}$.

This encryption scheme is well-known to be `IND-CCA` under the DDH assumption.

Labeled Cramer-Shoup Encryption on Vectors. The above scheme can be extended to encrypt vectors of group elements $\boldsymbol{M} = (M_1, \ldots, M_m) \in \mathbb{G}^m$: $m\text{-}\mathsf{MCS}^{\ell}_{\mathsf{pk}}(\boldsymbol{M}; \boldsymbol{r}) = (\mathsf{CS}^{\ell}_{\mathsf{pk}}(M_i; r_i))_i = (u_i = g^{r_i}, v_i = h^{r_i}, e_i = f^{r_i} \cdot M_i, w_i = (cd^{\theta})^{r_i})_i$, where $\theta = H(\ell, (u_i, v_i, e_i)_i)$, with indices range in

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1, \ldots, P_n$ via the following queries:

- **Upon receiving a query $(\texttt{NewSession}, \text{sid}, \text{ssid}, P_i, P_j, \pi)$ from party P_i:** Send $(\texttt{NewSession}, \text{sid}, \text{ssid}, P_i, P_j)$ to $\mathcal{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record $(\text{sid}, \text{ssid}, P_j, P_i, \pi')$, then record $(\text{sid}, \text{ssid}, P_i, P_j, \pi)$ and mark this record `fresh`.
- **Upon receiving a query $(\texttt{TestPwd}, \text{sid}, \text{ssid}, P_i, \pi')$ from the adversary $\mathcal{S}$:** If there is a record of the form (P_i, P_j, π) which is `fresh`, then do: If $pw = pw'$, mark the record `compromised` and reply to $\mathcal{S}$ with "correct guess". If $\pi \neq \pi'$, mark the record `interrupted` and reply with "wrong guess".
- **Upon receiving a query $(\texttt{NewKey}, \text{sid}, \text{ssid}, P_i, \mathsf{sk})$ from the adversary $\mathcal{S}$:** If there is a record of the form $(\text{sid}, \text{ssid}, P_i, P_j, \pi)$, and this is the first `NewKey` query for P_i, then:
  - If this record is `compromised`, or either P_i or P_j is corrupted, then output $(\text{sid}, \text{ssid}, \mathsf{sk})$ to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, π') with $\pi' = \pi$, and a key sk' was sent to P_j, and (P_j, P_i, π) was `fresh` at the time, then output $(\text{sid}, \text{ssid}, \mathsf{sk}')$ to P_i.
  - In any other case, pick a new random key sk' of length $\mathfrak{K}$ and send $(\text{sid}, \text{ssid}, sk')$ to P_i.

  Either way, mark the record $(\text{sid}, \text{ssid}, P_i, P_j, \pi)$ as `completed`.

Fig. 10. Ideal Functionality for PAKE $\mathcal{F}_{\text{pwKE}}$

$\{1, \ldots, m\}$. Such a ciphertext $C = (u_i, v_i, e_i, w_i)_i$ with label ℓ is decrypted by $M_i = e_i/u_i^z$, after having checked the validity of the ciphertext: $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$ for $i = 1, \ldots, m$.

This encryption scheme MCS is also `IND-CCA` under the DDH assumption. More precisely, if q_d is the number of decryption queries,

$$\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{ind-cca}}(m, q_d, t) \leq 2m \times \mathsf{Adv}_{\mathbb{G}}^{\texttt{ddh}}(t) + \mathsf{Succ}_{\mathcal{H}}^{\texttt{coll}}(t) + \frac{m(m+1)q_d}{p}.$$

Proof. Let us be given a tuple $(g, h, u, v) \in \mathbb{G}^4$. We choose random scalars $x_1, x_2, y_1, y_2, z \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, and set $\mathsf{sk} = (x_1, x_2, y_1, y_2, z)$ and $\mathsf{pk} = (g, h, c = g^{x_1}h^{x_2}, d = g^{y_1}h^{y_2}, f = g^z, H)$, where H is a random collision-resistant hash function from $\mathcal{H}$. We now consider an `IND-CCA` adversary on m-element vectors.

Game G_0: With the above setup, we play the `IND-CCA` game where the challenge ciphertext is on $\boldsymbol{M}$, that is chosen at random among the two vectors outputted by the adversary in the FIND stage.

Game G_1: In this game, instead of setting $f = g^z$, one sets $f = g^{z_1}h^{z_2}$, which essentially means that $z = z_1 + sz_2$, where $h = g^s$. Then, the decryption works as follows: $M_i = e_i/(u_i^{z_1}v_i^{z_2})$, after having checked the validity of the ciphertext: $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$ for $i = 1, \ldots, m$, relatively to the label ℓ.
One can note that the decryption algorithm yields the same result for correct ciphertexts (*i.e.*, ciphertexts such that, for $i = 1, \ldots, m$, $u_i = g^{r_i}$ and $v_i = h^{r_i}$ for some r_i): $u_i^{z_1}v_i^{z_2} = (g^{r_i})^{z_1}(h^{r_i})^{z_2} = (g^{r_i})^{z_1+sz_2} = u_i^z$. Let us show that any adversary (even unbounded) cannot generate an incorrect ciphertext which passes the validity test $w_i \stackrel{?}{=} u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$, for all i, with non-negligible probability. Let us indeed consider such incorrect ciphertext. For some index i, $w_i = u_i^{x_1+\theta y_1} v_i^{x_2+\theta y_2}$, but $u_i = g^{r_i}$ and $v_i = h^{r'_i}$ with $r_i \neq r'_i$. By taking the discrete logarithm in base g, and by setting $\delta_i = r'_i - r_i$, we get:

$$\log c = x_1 + sx_2 \qquad \log d = y_1 + sy_2$$
$$\log w_i = r_i(x_1 + \theta y_1) + sr'_i(x_2 + \theta y_2) = r_i(\log c + \theta \log d) + s\delta_i(x_2 + \theta y_2).$$

Since c and d do not reveal any information about x_2 and y_2, $s\delta_i(x_2+\theta y_2)$ is unpredictable and thus the correct value for w_i is unpredicatble too. An incorrect ciphertext is declared valid with probability less than m/p, and thus the distance between the two games is bounded by mq_d/p, where q_d is the number of decryption queries.

Game $\boldsymbol{G}_2$: In this game the challenge ciphertext is generated following the new decryption approach: $C^* = (u_i^*, v_i^*, e_i^* = u_i^{*z_1} v_i^{*z_2} \cdot M_i, w_i^* = u_i^{*x_1+\theta^* y_1} v_i^{*x_2+\theta^* y_2})_i$, where we have the equality $\theta^* = H(\ell^*, (u_i^*, v_i^*, e_i^*)_i)$, with all the tuples (g, h, u_i^*, v_i^*) being DH tuples. As already shown above, since truly DH tuples are used, this makes no difference.

Game $\boldsymbol{G}_3$: In this game, the challenge ciphertext $C^* = (u_i^*, v_i^*, e_i^*, w_i^*)_i$, uses tuples $(u_i^*, v_i^*, e_i^*, w_i^*)$ randomly and independently chosen in $\mathbb{G}^4$.

In order to bound the distance between the two games, we use a sequence of hybrid games: in G_j, for $i \le j$, (g, h, u_i^*, v_i^*) are DH tuples and $e_i^* = u_i^{*z_1} v_i^{*z_2} \cdot M_i$, $w_i^* = u_i^{*x_1+\theta^* y_1} v_i^{*x_2+\theta^* y_2}$, where $\theta^* = H(\ell, (u_i^*, v_i^*, e_i^*)_i)$, while for $i > j$, tuples $(u_i^*, v_i^*, e_i^*, w_i^*)$ are randomly and independently chosen in $\mathbb{G}^4$. One can note that G_0 is exactly $\boldsymbol{G}_3$, while G_m is $\boldsymbol{G}_2$.

Let us modify a little bit G_j into G'_j, in which (g, h, u, v) is a DH tuple and $u_j^* = u$, $v_j^* = v$, $e_j^* = u_j^{*z_1} v_j^{*z_2} \cdot M_j$, $w_j^* = u_j^{*x_1+\theta^* y_1} v_j^{*x_2+\theta^* y_2}$. This does not change anything. We now alter it into G''_j, where (g, h, u, v) is a random tuple. In such a case, $e_j^* = u^{z_1} v^{z_2} \cdot M_j = u^z h^{\delta z_2} \cdot M_j$, where $\delta = r' - r$, with $u = g^r$ and $v = h^{r'}$, and thus $\delta \neq 0$ with overwhelming probability, as well as $z_2 \neq 0$. In addition, z_2 is totally unpredictable, unless some incorrect ciphertexts are decrypted.

As a conclusion, unless some incorrect ciphertexts are decrypted, e_j^* is totally unpredictable. But an incorrect ciphertext $C = (u_i, v_i, e_i, w_i)_i$ is decrypted if for some i

$$\log c = x_1 + sx_2 \qquad \log d = y_1 + sy_2$$
$$\log w_j^* = r(x_1 + \theta^* y_1) + sr'(x_2 + \theta^* y_2)$$
$$\log w_i = r_i(x_1 + \theta y_1) + sr'_i(x_2 + \theta y_2)$$

whereas $\delta_i = r'_i - r_i \neq 0$. The determinant of the system is $s^2 \delta \delta_i (\theta^* - \theta)$, which is non-zero, unless $\theta^* = \theta$. But two cases only are possible:

– if $(\ell, (u_i, v_i, e_i)_i) \neq (\ell^*, (u_i^*, v_i^*, e_i^*)_i)$, $\theta^* = \theta$ leads to a collision for H;
– if $(\ell, (u_i, v_i, e_i)_i) = (\ell^*, (u_i^*, v_i^*, e_i^*)_i)$, then the query is not allowed.

Therefore, the determinant being non-zero, w_i is unpredictable and thus the probability that at least one incorrect ciphertext is declared valid is bounded by mq_d/p.

In addition, w_j^* is also unpredictable. This means that G''_j is exactly G_{j-1}. but the distance between the two games is bounded by $mq_d/p + 2\mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t)$, assuming no collision for H. Eventually, the distance between $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ is bounded by $m^2 q_d/p + 2m\mathsf{Adv}^{\mathtt{ddh}}_{\mathbb{G}}(t) + \mathsf{Succ}^{\mathtt{coll}}_{\mathcal{H}}(t)$.

This concludes the proof since in the last game, the values e_i^* are independent of the message, and thus of the bit involved in the security game: the advantage of the adversary is 0. □

We now provide a stronger security notion for encryption of vectors, which is useful for our application to $\mathtt{E}^2$-commitments.

B.3 Vector-Indistinguishability with Partial Opening, under Chosen-Ciphertext Attacks

New Security Notion. In our applications, when encrypting vectors, the adversary will get some of the random coins used for encryption. We thus define a stronger security notion:

Vector-indistinguishability with partial opening, under chosen-ciphertext attacks: this security notion VIND-PO-CCA can be formalized by the following experiments $\mathsf{Exp}_{\mathcal{A}}^{\texttt{vind-po-cca-}b}(\mathfrak{K})$, where the adversary $\mathcal{A}$ keeps some internal state between the various calls FIND and GUESS, and makes use of the above ODecrypt oracle. However, $\mathsf{Encrypt}^*$ has an additional input $\varDelta$, that consists of the common values in $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$, and $\perp$ at the places of distinct values. It also outputs the values $\boldsymbol{r}$ that allow to check that C^* actually encrypts a vector $\boldsymbol{M}$ that corresponds to $\varDelta$ (*i.e.*, that is equal to $\varDelta$ for places different than $\perp$). The exact definition of these values $\boldsymbol{r}$ depend on the actual encryption scheme.

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{vind-po-cca-}b}(\mathfrak{K})$
    $\texttt{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$
    $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}(\texttt{param})$
    $(\ell^*, \boldsymbol{M}_0, \boldsymbol{M}_1, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\texttt{FIND} : \mathsf{pk})$
    $\varDelta = \boldsymbol{M}_0 \cap \boldsymbol{M}_1$
    $(C^*, \boldsymbol{r}) \xleftarrow{\$} \mathsf{Encrypt}^{*\ell^*}(\mathsf{pk}, \varDelta, \boldsymbol{M}_b)$
    $b' \xleftarrow{\$} \mathcal{A}^{\mathsf{ODecrypt}^{\cdot}(\cdot)}(\mathsf{state}, \texttt{GUESS} : C^*, \boldsymbol{r})$
    if $(\ell^*, C^*) \in \texttt{CTXT}$ then return 0
    else return $b'$
```

This models the fact that when distinct random coins are used for each components of the vector, the random coins of the common components can be revealed, it should not help to distinguish which vector has been encrypted.

These experiments $\mathsf{Exp}_{\mathcal{A}}^{\texttt{vind-po-cca-}b}(\mathfrak{K})$ define the two advantages $\mathsf{Adv}_{\mathcal{E}}^{\texttt{vind-po-cca}}(\mathcal{A}, \mathfrak{K})$ and $\mathsf{Adv}_{\mathcal{E}}^{\texttt{vind-po-cca}}(t)$. As above, we will use $\mathsf{Adv}_{\mathcal{E}}^{\texttt{vind-po-cca}}(m, q_d, \gamma, t)$ to make precise the length m of the vectors, and to bound by q_d the number of decryption queries and by γ the number of distinct values in the pairs of vectors.

Labeled Cramer-Shoup Encryption on Vectors. For the Cramer-Shoup encryption on vectors MCS, the values $\boldsymbol{r}$ output by $\mathsf{Encrypt}^*$ are the random coins r_i corresponding to to the common components of $\boldsymbol{M}_0$ and $\boldsymbol{M}_1$ (*i.e.*, for i such that $M_{0,i} = M_{1,i} = \varDelta_i$). These values are sufficient to check that C^* actually encrypts a vector corresponding to $\varDelta$.

We can prove that MCS is VIND-PO-CCA using a slight variant of the IND-CCA proofs given in Section B.2. More precisely, we use the same games except that for i such that $M_{0,i} = M_{1,i}$, in Games $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$, we compute u_i^* and v_i^* as in Game $\boldsymbol{G}_1$: $u_i^* = g^{r_i}$ and $v_i^* = h^{r_i}$ for a random scalar r_i. The hybrid technique to prove the indistinguishability of $\boldsymbol{G}_2$ and $\boldsymbol{G}_3$ uses γ steps only (instead of m steps), with γ the number of distinct components. Finally, we get:

$$\mathsf{Adv}_{\mathsf{MCS}}^{\texttt{vind-po-cca}}(m, q_d, \gamma, t) \leq 2\gamma \times \mathsf{Adv}_{\mathbb{G}}^{\texttt{ddh}}(t) + \mathsf{Succ}_{\mathcal{H}}^{\texttt{coll}}(t) + \frac{m(\gamma+1)q_d}{p}.$$

C Useful SPHFs

In this appendix, we show three constructions of SPHF for $\mathcal{E}^2\mathcal{C}$ and summarize their costs. We use the framework from [BBC+13].

C.1 Construction of the SPHFs

The language we are interested in is the language of valid commitments $C = (\boldsymbol{a}, \boldsymbol{b}) \in \mathbb{G}_2^m \times \mathbb{G}_1^{8m}$ under label ℓ of some fixed vector $\boldsymbol{M}$, where the witness is the decommitment information $\delta = \boldsymbol{s} = (s_{i,M_i})_i \in \mathbb{Z}_p^m$. More precisely, a word C is in the language if and only if there exists δ and $\boldsymbol{d} = (d_{i,M_i})_i \in \mathbb{G}_1^m$ such that, for all $i = 1, \ldots, m$:

$$b_{i,M_i} = \mathsf{CS}_{\mathsf{pk}}^{\ell'}(d_{i,M_i}, \theta; s_{i,M_i}) \qquad \text{and} \qquad e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2),$$

with $\ell' = (\ell, \boldsymbol{a})$ and $\theta = H(\ell', (u_{i,j}, v_{i,j}, e_{i,j})_{i,j})$. If we set $b_{i,j} = (u_{i,j}, v_{i,j}, e_{i,j}, w_{i,j})_{i,j}$, we can write the previous conjunction as: for all $i = 1, \ldots, m$,

$$(u_{i,M_i}, v_{i,M_i}, e_{i,M_i}, w_{i,M_i}) = (g_1^{s_{i,M_i}}, h_1^{s_{i,M_i}}, f_1^{s_{i,M_i}} \cdot d_{i,M_i}, (cd^{\theta})^{s_{i,M_i}})$$
$$\text{and} \qquad e(g_1, a_i/T^{M_i}) = e(d_{i,M_i}, g_2).$$

Since e is non-degenerated, we finally remark that we can eliminate the need of d_{i,M_i}, by lifting everything in $\mathbb{G}_T$, and checking that, first, the ciphertexts are all valid and, second, the plaintexts satisfy the appropriate relations:

$$(e(u_{i,M_i}, g_2), e(v_{i,M_i}, g_2), e(w_{i,M_i}, g_2)) = (e(g_1^{s_{i,M_i}}, g_2), e(h_1^{s_{i,M_i}}, g_2), e((cd^\theta)^{s_{i,M_i}}, g_2))$$
$$e(e_{i,M_i}, g_2) = e(f_1^{s_{i,M_i}}, g_2) \cdot e(g_1, a_i/T^{M_i}).$$

From these expressions we can derive three SPHFs.

KV-SPHF. A KV-SPHF is a SPHF for which hp does not depend on C and the smoothness holds even if the adversary can see hp before choosing C (see [BBC+13] for a precise definition). Let us first show how to construct a KV-SPHF checking the previous condition for only a fixed index i. For this purpose, we use the following matrices (for the framework of [BBC+13]:

$$\Gamma = \begin{pmatrix} g_1 & 0 & h_1 & f_1 & c \\ 0 & g_1 & 0 & 0 & d \end{pmatrix} \tag{2}$$
$$\boldsymbol{\lambda}_i = (g_2^{s_{i,M_i}}, g_2^{s_{i,M_i}\theta})$$
$$\boldsymbol{\lambda}_i \cdot \Gamma = (e(g_1^{s_{i,M_i}}, g_2), e(g_1^{s_{i,M_i}\theta}, g_2), e(h_1^{s_{i,M_i}}, g_2), e(f_1^{s_{i,M_i}}, g_2), e((cd^\theta)^{s_{i,M_i}}, g_2))$$
$$\Theta_i(C) = (e(u_{i,M_i}, g_2), e(u_{i,M_i}^\theta, g_2), e(v_{i,M_i}, g_2), e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}), e(w_{i,M_i}, g_2))$$

With $\mathsf{hk} = (\eta_1, \eta_2, \alpha, \beta, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, we get $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} h_1^\alpha f_1^\beta c^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d^\mu) \in \mathbb{G}_1^2$.

Eventually, to check that C actually commits to $\boldsymbol{M} = (M_i)_i$, we can define a KV-SPHF, as follows:

$$\mathsf{hk} = (\eta_{i,1}, \eta_{i,2}, \alpha_i, \beta_i, \mu_i)_i \xleftarrow{\$} \mathbb{Z}_p^{5m} \quad \mathsf{hp} = (\mathsf{hp}_{i,1} = g_1^{\eta_{i,1}} h_1^{\alpha_i} f_1^{\beta_i} c^{\mu_i}, \mathsf{hp}_{i,2} = g_1^{\eta_{i,2}} d^{\mu_i})_i \in \mathbb{G}_1^{2m},$$
$$H = \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \stackrel{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^{(\eta_{i,1}+\theta\eta_{i,2})} \cdot v_{i,M_i}^{\alpha_i}, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\beta_i} \cdot e(w_{i,M_i}^{\mu_i}, g_2) \right)$$
$$= e(\textstyle\prod_i (\mathsf{hp}_{i,1}\mathsf{hp}_{i,2}^\theta)^{s_{i,M_i}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, (s_{i,M_i})_i) = H'.$$

Since $\delta = (s_{i,M_i})_i$, this will allow the one-round PAKE, described in Section 6.

CS-SPHF. When C is sent before hp, we can use a CS-SPHF instead of a KV-SPHF (for which the smoothness holds only when the adversary cannot see hp before choosing C). Here is a more efficient CS-SPHF, with a common projection key for all the components, but an additional random ε[1]:

$$\Gamma = \begin{pmatrix} g_1 & 0 & h_1 & f_1 & c \\ 0 & g_1 & 0 & 0 & d \end{pmatrix} \qquad \boldsymbol{\lambda} = \left(\textstyle\prod_i g_2^{s_{i,M_i}\varepsilon^{i-1}}, g_2^{s_{i,M_i}\theta\varepsilon^{i-1}} \right)$$
$$\boldsymbol{\lambda} \cdot \Gamma = \Big(e(\textstyle\prod_i g_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2), \quad e(\prod_i g_1^{s_{i,M_i}\theta\varepsilon^{i-1}}, g_2), \quad e(\prod_i h_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2),$$
$$e(\textstyle\prod_i f_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2), \quad e(\prod_i (cd^\theta)^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \Big)$$
$$\Theta(C) = \Big(\textstyle\prod_i e(u_{i,M_i}^{\varepsilon^{i-1}}, g_2), \quad e(\prod_i u_{i,M_i}^{\theta\varepsilon^{i-1}}, g_2), \quad e(\prod_i v_{i,M_i}^{\varepsilon^{i-1}}, g_2),$$
$$\textstyle\prod_i (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^{\varepsilon^{i-1}}, \quad e(\prod_i w_{i,M_i}^{\varepsilon^{i-1}}, g_2) \Big),$$

which leads to

$$\mathsf{hk} = (\eta_1, \eta_2, \alpha, \beta, \mu, \varepsilon) \xleftarrow{\$} \mathbb{Z}_p^6 \quad \mathsf{hp} = (\varepsilon, \mathsf{hp}_1 = g_1^{\eta_1} h_1^\alpha f_1^\beta c^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d^\mu) \in \mathbb{Z}_p \times \mathbb{G}_1^2,$$
$$H = \mathsf{Hash}(\mathsf{hk}, \boldsymbol{M}, C) \stackrel{\text{def}}{=} \prod_i \left(e(u_{i,M_i}^{(\eta_1+\theta\eta_2)} \cdot v_{i,M_i}^\alpha, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^\beta \cdot e(w_{i,M_i}^\mu, g_2) \right)^{\varepsilon^{i-1}}$$
$$= e(\textstyle\prod_i (\mathsf{hp}_1\mathsf{hp}_2^\theta)^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{M}, C, (s_{i,M_i})_i) = H'.$$

[1] Actually, it is possible to choose $\varepsilon \xleftarrow{\$} \{0,1\}^{\mathfrak{K}}$.

More precisely, following the framework from [BBC+13], since hp is not known at the time C is generated, to prove that the resulting CS-SPHF is smooth, we just need to prove that for any invalid C (not in the language), the probability that $\Theta(C)$ is not a *linear* combination of the rows of $\Gamma(C)$ is overwhelming, over the random choice of ε. Indeed, if $\Theta(C)$ is independent of rows of $\Gamma(C)$, H is completely unpredictable even given hp.

Let us indeed consider an invalid commitment C, and let us write $b_{i,M_i} = (u_{i,M_i}, v_{i,M_i}, e_{i,M_i}, w_{i,M_i})_i = (g_1^{s_i}, h_1^{t_i}, f_1^{s'_i} \cdot d_{i,j}, (cd^\theta)^{s''_i})_i$, with d_{i,M_i} such that $e(d_{i,M_i}, g_2) = e(g_1, a_i/T^{M_i})$, which is always possible if we suppose g_1, h_1, f_1 and cd^θ are all generators[2]. Then, since C is invalid, there exists some i^*, such that either $t_{i^*} \neq s_{i^*}$ or $s'_{i^*} \neq s_{i^*}$ or $s''_{i^*} \neq s_{i^*}$. Let us suppose $t_{i^*} \neq s_{i^*}$. The other cases are similar. Then we remark that for $\Theta(C)$ to be linearly dependent of rows of Γ, it is necessary that $e(\prod_i v_{i,M_i}^{\varepsilon^{i-1}}, g_2) = e(h_1^{s_i \sum_i \varepsilon^{i-1}}, g_2)$, since the first coefficient of the linear combination is necessary $\lambda_1 = g_2^{\sum_i s_i \varepsilon^{i-1}} = \prod_i g_2^{s_i \varepsilon^{i-1}}$. This implies that: $\sum_i t_i \varepsilon^{i-1} = \sum_i s_i \varepsilon^{i-1}$, by looking at the discrete logarithm in base $e(g_1, g_2)$. In other words ε has to be a root of the m-degree polynomial $\sum_i (t_i - s_i) X^i$, which is not null because $t_{i^\star} \neq s_{i^\star}$. This polynomial has at most m roots, and so at most m distinct ε (among the p possible ε in $\mathbb{Z}_p$) lead to a $\Theta(C)$ linearly dependent of rows of Γ. Finally, we conclude that, with probablity at least $1 - m/p$, $\Theta(C)$ is independent of rows of Γ. This proves the smoothness of the SPHF.

GL-SPHF. When C is sent in advance, and thus known when generating hp, as in the Oblivious Transfer protocol described in Section 7, we can use a more efficient GL-SPHF instead of a CS-SPHF:

$$\begin{aligned}
\Gamma(C) &= \big(g_1\ h_1\ f_1\ (cd^\theta)\big) \\
\boldsymbol{\lambda}_i &= g_2^{s_{i,M_i}} \\
\boldsymbol{\lambda}_i \cdot \Gamma(C) &= (e(g_1^{s_{i,M_i}}, g_2), e(h_1^{s_{i,M_i}}, g_2), e(f_1^{s_{i,M_i}}, g_2), e((cd^\theta)^{s_{i,M_i}}, g_2)) \\
\Theta_i(C) &= (e(u_{i,M_i}, g_2), e(v_{i,M_i}, g_2), e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}), e(w_{i,M_i}, g_2))
\end{aligned}$$

and:

$$\begin{aligned}
&\mathsf{hk} = (\eta, \alpha, \beta, \mu, \varepsilon) \xleftarrow{\$} \mathbb{Z}_p^5 \quad \mathsf{hp} = (\varepsilon, \mathsf{hp}_1 = g_1^\eta h_1^\alpha f_1^\beta (cd^\theta)^\mu) \in \mathbb{Z}_p \times \mathbb{G}_1, \\
H = \mathsf{Hash}(\mathsf{hk}, M, C) &\stackrel{\text{def}}{=} \prod_i \Big(e(u_{i,M_i}^\eta \cdot v_{i,M_i}^\alpha, g_2) \cdot (e(e_{i,M_i}, g_2)/e(g_1, a_i/T^{M_i}))^\beta \cdot e(w_{i,M_i}^\mu, g_2)\Big)^{\varepsilon^{i-1}} \\
&= e(\textstyle\prod_i \mathsf{hp}_1^{s_{i,M_i}\varepsilon^{i-1}}, g_2) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, (s_{i,M_i})_i) = H'.
\end{aligned}$$

C.2 Complexity

In Table 3, we summarize the cost of the various SPHFs. We remark that in the CS-SPHF and GL-SPHF, if $m = 1$, we do not need ε. That is why this case is handled separately. In all cases, the hash value consists of 1 group element in $\mathbb{G}_T$. In practice we use an entropy extractor on this hash value.

D Security Proofs

Each flow in the concrete protocols should include the tuple $(\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, but we omit it for the sake of simplicity. This tuple is needed for the queries the simulator to asks to the ideal functionality (in the description of the ideal games at the end of each sequence of games below).

[2] It is easy to extend to the case where some of them are not generators (*i.e.*, are equal to 1).

Table 3. Cost of the three SPHFs for $\mathcal{E}^2\mathcal{C}$

	hk	hp
KV-SPHF	$5m \times \mathbb{Z}_p$	$2m \times \mathbb{G}_1$
CS-SPHF (for $m = 1$)	$5 \times \mathbb{Z}_p$	$2 \times \mathbb{G}_1$
CS-SPHF (for $m \geq 2$)	$6 \times \mathbb{Z}_p$	$2 \times \mathbb{G}_1 + 1 \times \mathbb{Z}_p$
GL-SPHF (for $m = 1$)	$4 \times \mathbb{Z}_p$	$1 \times \mathbb{G}_1$
GL-SPHF (for $m \geq 2$)	$5 \times \mathbb{Z}_p$	$1 \times \mathbb{G}_1 + 1 \times \mathbb{Z}_p$

D.1 Proof of Theorem 4 (UC-Secure Commitment from a Labeled $\mathtt{E}^2$-Commitment)

We first prove this theorem in the case the labeled $\mathtt{E}^2$-commitment scheme $\mathcal{C}$ is additionally strongly-binding-extractable, and explain afterwards the difference when the commitment is strong-simulation-indistinguishable.

With Strong-Binding-Extractability. We thus exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$.

Essentially, one first makes the setup algorithm additionally output the trapdoor (setup-indistinguishability); one can then replace all the commitment queries by simulated (fake) commitments (simulation-indistinguishability). Eventually, when simulating the receiver, the simulator extracts the committed value x (using ExtCom), which should be the same as the one that will be open later (strong-binding-extractability). One can then split the simulation, to open it when required only with the appropriate information: the committed value sent be the environment is not required anymore. More details follow:

Game $\boldsymbol{G}_0$: This is the real game.

Game $\boldsymbol{G}_1$: In this game, the simulator generates correctly every flows from the honest players, as they would do themselves, knowing the inputs x sent by the environment to the senders. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\boldsymbol{G}_2$: In this game, we just replace the setup algorithm $\mathsf{SetupCom}$ by $\mathsf{SetupComT}$ that additionally outputs the trapdoor $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under *setup indistinguishability*. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: We first deal with honest senders: we replace all the commitments $(C, \delta) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(x)$ with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ in Step 1. of the commit phase of honest players by simulated commitments $(C, \delta) \overset{\$}{\leftarrow} \mathsf{SCom}^{\ell}(\tau, x)$, which means $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$. We then store (ℓ, x, C) in Λ.
With an hybrid proof, applying the $\mathsf{Exp}^{\mathtt{sim\text{-}ind}}$ security game for each step, in which SCom is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games. In case of corruption of the sender, one learns the already known value x.

Game $\boldsymbol{G}_4$: We now deal with honest receivers: when receiving a fresh commitment C from the adversary or a replay from another session (and thus under a different label), the simulator extracts the committed value x. If the adversary later opens to a different value at the decommit phase, the simulator rejects it. If it was accepted in the previous game, then one breaks the strong-binding-extractability.
More generally, the trapdoor correctness ensures that valid decommitments (accepted in the previous game) will be accepted in this game too, which makes almost no difference between the two games, but with probability bounded by $\mathsf{Succ}_{\mathcal{C}}^{\mathtt{s\text{-}bind\text{-}ext}}(t)$. Note that in

the experiment $\mathsf{Exp}^{\texttt{s-bind-ext}}$, the adversary has access to both the fake commitment oracle (SCom) and the extraction oracle (ExtCom), which are indeed required here.

Game $\boldsymbol{G}_5$: We do not use anymore the knowledge of x: the simulator knows the trapdoor τ and generates $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, to send C during the commit phase of honest players. When receiving a `Revealed`-message on x, it then generates $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ to actually open the commitment. We essentially break the atomic SCom in the two separated processes SimCom and $\mathsf{OpenCom}$. This does not change anything from the previous game except that Λ is first filled with $(\ell, \perp, C)$ at the commit time and then updated to (ℓ, x, C) at the opening time. In case of corruption of the sender, one learns the committed value that is thereafter used at the decommit phase for x.

Game $\boldsymbol{G}_6$: We can now make use of the functionality, which leads to the following simulator:

- when receiving a commitment C from the adversary, and thus either freshly generated by the adversary or a replay of a commitment C generated by the simulator in another session (with a different label), the simulator extracts the committed value x, and uses it to send a `Commit` message to the ideal functionality. A dummy value is used in case of bad extraction;
- when receiving a `Receipt`-message, which means that an honest player has committed a value, the simulator generates $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, to send C during the commit phase of the honest player;
- when receiving (x, δ), if the verification succeeds, the simulator asks for a `Reveal` query to the ideal functionality;
- when receiving a `Revealed`-message on x, it then generates $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ to actually open the commitment.

Any corruption just reveals x earlier, which allows a correct simulation of the opening.

With Strong-Simulation-Indistinguishability. In the other case, where the labeled $\mathtt{E}^2$-commitment is additionally strongly-simulation-indistinguishable, we can just first simulate the receiver before modifying the simulation of the sender with fake commitments: one will first apply extractability and then strong-simulation-indistinguishability. This concretely means that we swap $\boldsymbol{G}_3$ and $\boldsymbol{G}_4$. This leads to the same simulator in $\boldsymbol{G}_6$.

Cost of the Reductions. More precisely, for any environment, its advantage in distinguishing the ideal world (with the ideal functionality from Figure 4) and the real world (with the commitment scheme $\mathcal{C}$) is bounded by both $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q_s \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{sim-ind}}(t) + \mathsf{Succ}_{\mathcal{C}}^{\texttt{s-bind-ext}}(t)$ and $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q_s \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) + \mathsf{Succ}_{\mathcal{C}}^{\texttt{bind-ext}}(t)$, where q_s is the number of concurrent sessions and t its running time.

D.2 Proofs for Sections 4.3 and 5.4 (Security Properties of our $\mathcal{E}^2\mathcal{C}$ Commitment)

Setup-indistinguishability. this is trivially satisfied since the two setup algorithms are exactly the same but just output the trapdoor or not, and thus $\mathsf{Adv}_{\mathcal{E}^2\mathcal{C}}^{\texttt{setup-ind}}(t) = 0$ for any t.

(t, ε)-strong-simulation-indistinguishability. Let us build a sequence of games from the security experiment with $b = 1$ to the experiment with $b = 0$. We stress that SCom does not only output $C = (\boldsymbol{a}, \boldsymbol{b})$, but also $\delta = ((d_{i,M_i}, s_{i,M_i})_i)$, where the $s_{i,j}$'s are the random coins in the multi-Cramer-Shoup encryption.

1. We first start with the real game with $b = 1$ (use of SCom for the challenge commitment), with all the trapdoors to emulate the oracles;
2. the simulator now knows the equivocation trapdoor to emulate the SCom-oracle, but has just access to the decryption oracle to emulate the ExtCom-oracle;
3. for the challenge oracle on $x = (x_i)_i$, the simulator uses $r_{i,1-x_i} = 0$, which leads to the plaintexts $d_{i,1-x_i} = 1$ that are thereafter encrypted under the Cramer-Shoup encryption

scheme. Applying the VIND-PO-CCA security of the MCS encryption scheme, in which the m components of the vector that correspond to the committed vector x are the same in the two $2m$-long vectors, one can note that the bias is upper-bounded by $\mathsf{Adv}_{\mathsf{MCS}}^{\mathtt{vind\text{-}po\text{-}cca}}(2m, q_d, m, t)$, where q_d the number of extraction queries. The two vectors submitted to the encryption oracle $\mathsf{Encrypt}^*$ in the security game VIND-PO-CCA are $(d_{1,0}, d_{1,1}, \ldots, d_{m,0}, d_{m,1})$, where the d_{i,x_i}'s keep the same in the two games, but the $d_{i,1-x_i}$'s are all replaced by 1 in the second game. Then, the $\mathsf{Encrypt}^*$ oracle additionally outputs the s_{i,x_i}'s (that correspond to the common components), which allows to output δ.

4. giving back all the trapdoors to the simulator, we are in the real game with $b = 0$ (use of Com for the challenge commitment).

In conclusion, one thus gets $\mathsf{Adv}_{\mathcal{E}^2\mathcal{C}}^{\mathtt{s\text{-}sim\text{-}ind}}(t) \leq \mathsf{Adv}_{\mathsf{MCS}}^{\mathtt{vind\text{-}po\text{-}cca}}(2m, q_d, m, t)$.

Strong binding extractability. Let us build a sequence of games from the security experiment to an attack to the DDH in $\mathbb{G}_2$.

1. we first start with the real game, with all the trapdoors to emulate the oracles;
2. the simulator replaces all the SCom-oracle queries by Com-oracle queries. With an hybrid proof, where we replace sequentially the SCom emulations by Com emulations, as above, one introduces a bias upperbounded by $q_c \cdot \mathsf{Adv}_{\mathcal{E}^2\mathcal{C}}^{\mathtt{s\text{-}sim\text{-}ind}}(t)$, and thus $q_c \cdot \mathsf{Adv}_{\mathsf{MCS}}^{\mathtt{vind\text{-}po\text{-}cca}}(2m, q_d, m, t)$, where q_c is the number of SCom-queries and q_d the number of extract queries;
3. the simulator does not need any more the equivocation trapdoor, but can still extract the correct d_{i,x_j}, by decrypting the Cramer-Shoup ciphertexts, to open the commitment with $e(g_1, a_i/T^{x_i}) = e(d_{i,x_i}, g_2)$. When the adversary breaks the strong-binding-extractability, it provides C with a valid opening $(\boldsymbol{M}, \delta)$, whereas C extracts to $\boldsymbol{M'} \neq \boldsymbol{M}$ (possibly $\perp$).
   Since opening/verification in one way is possible $\boldsymbol{M}$, this means that the Cramer-Shoup decryption gives at least one valid opening for each a_i. But because of the different extraction output $\boldsymbol{M'}$, extraction technique is ambiguous on C: for an index i, it can provide two different opening values for a_i, which breaks the DDH assumption in $\mathbb{G}_2$.

In conclusion, one thus gets $\mathsf{Succ}_{\mathcal{E}^2\mathcal{C}}^{\mathtt{s\text{-}bind\text{-}ext}}(t) \leq q_c \cdot \mathsf{Adv}_{\mathsf{MCS}}^{\mathtt{vind\text{-}po\text{-}cca}}(2m, q_d, m, t) + \mathsf{Adv}_{\mathbb{G}_2}^{\mathsf{ddh}}(t)$, where q_c is the number of SCom-queries and q_d the number of extract queries.

Robustness. In the above proof of strong-binding-extractability, as soon as different opening values exist, by decrypting the Cramer-Shoup ciphertexts, one breaks the DDH assumption in $\mathbb{G}_2$: $\mathsf{Succ}_{\mathcal{E}^2\mathcal{C}}^{\mathtt{robust}}(t) \leq q_c \cdot \mathsf{Adv}_{\mathsf{MCS}}^{\mathtt{ind\text{-}cca}}(t) + \mathsf{Adv}_{\mathbb{G}_2}^{\mathsf{ddh}}(t)$, where q_c is the number of SCom-queries.

Strong pseudo-randomness. For the sake of simplicity, we write $x = \boldsymbol{M}$ and $x' = \boldsymbol{M'}$. We also write: $C = (\boldsymbol{a}, \boldsymbol{b})$ and $C' = (\boldsymbol{a'}, \boldsymbol{b'})$. To prove the strong pseudo-randomness, we use the following sequence of games:

Game $\boldsymbol{G}_0$: This game is the experiment $\mathsf{Exp}_{\mathcal{A}}^{\mathtt{c\text{-}s\text{-}ps\text{-}rand\text{-}0}}$.

Game $\boldsymbol{G}_1$: In this game, before computing H', we compute $\boldsymbol{M'} \leftarrow \mathsf{ExtCom}^{\ell'}(\tau, C')$ and we abort if for some i, the decryptions of $b'_{i,0}$ and $b'_{i,1}$ give valid opening values of a'_i for 0 and 1 respectively. In other words, if C' is not perfectly binding, we abort.[3]
This game is indistinguishable from the previous one, using the proof of robustness.

Game $\boldsymbol{G}_2$: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H' by a random value.
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, the fact that $\boldsymbol{M'} \neq \boldsymbol{M}$ and C' is perfectly binding (otherwise, we would have aborted), so that $(\ell', C') \notin L_{\boldsymbol{M}}$, and thanks to the fact that H could have been computed as follows: $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, \boldsymbol{M})$ and $H \leftarrow \mathsf{ProjHash}(\mathsf{hp}, L_{\boldsymbol{M}}, (\ell, C), \delta)$.

[3] Actually, we may abort more often than that, but at least, if the commitment C' is honestly generated, we do not abort.

Game $\boldsymbol{G}_3$: In this game, we replace $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$ by $C \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(\boldsymbol{M''})$ for some arbitrary $\boldsymbol{M''} \neq \boldsymbol{M}$. This game is indistinguishable thanks to strong simulation indistinguishability (since eqk is not used, SimCom could have been replaced by SCom with a $\boldsymbol{M''}$ as message).

Game $\boldsymbol{G}_4$: In this game, when $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H by a random value.
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, and the fact that C is a real commitment of $\boldsymbol{M''} \neq \boldsymbol{M}$ and so that $(\ell, C) \notin L_{\boldsymbol{M}}$.
Notice that we could not have done this if $\boldsymbol{M'} = \boldsymbol{M}$, since, in this case, we still need to use hk to compute the hash value H' of C'. We are handling this (tricky) case in the following game.

Game $\boldsymbol{G}_5$: In this game, we replace H by a random value, in the case $\boldsymbol{M'} = \boldsymbol{M}$. So now H will be completely random, in all cases (since it was already the case when $\boldsymbol{M'} \neq \boldsymbol{M}$).
Let $\theta = H(\ell, (u_{i,j}, v_{i,j}, e_{i,j})_{i,j})$ and $\theta' = H(\ell', (u'_{i,j}, v'_{i,j}, e'_{i,j})_{i,j})$. Finally, we write $s'_{i,j} = \log u'_{i,j}$ for all i, j, log being the discrete logarithm in base g_1. There are two cases:

1. for all i, $v'_{i,M_i} = h_1^{s'_{i,M_i}}$. In this case, since C' extracts to $\boldsymbol{M}$, this means that
$$w'_{i,M_i} = u'^{x_1+\theta' y_1}_{i,M_i} \cdot v'^{x_2+\theta' y'_2}_{i,M_i},$$
and so from the definition of c and d, we have that:
$$w'_{i,M_i} = (c \cdot d^{\theta'})^{s'_{i,M_i}}.$$
This means that $(\ell', C') \in L_{\boldsymbol{M}}$, and its hash value H' could be computed knowing only hp and $(s'_{i,M_i})_i$. Therefore, the hash value H of C looks random by smoothness.
2. for some i, $v'_{i,M_i} \neq h_1^{s'_{i,M_i}}$. Then since $v_{i,M_i} = h_1^{s_{i,M_i}}$, the rows of the matrix Γ in Equation 2 (page 25) and the two vectors $\Theta_i(C)$ and $\Theta_i(C')$ are linearly independent. Then, even given access to the hash value H' of C' and the projection key hp, the hash value H of C looks perfectly random.

The following games are just undoing the modifications we have done, but keeping H picked at random

Game $\boldsymbol{G}_6$: In this game, we now compute C as originally using SimCom. This game is indistinguishable thanks to strong simulation indistinguishability.

Game $\boldsymbol{G}_7$: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we compute H' as originally (as the hash value of C'). This game is indistinguishable from the previous one thanks to the smoothness of the SPHF.

Game $\boldsymbol{G}_8$: In this game, we do not extract $\boldsymbol{M'}$ from C' nor abort when C' is not perfectly binding. Thanks to the robustness, this game is indistinguishable from the previous one.
We remark that this game is exactly the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-1}}$.

Finally, one thus gets $\mathsf{Succ}^{\texttt{c-s-ps-rand}}_{\mathcal{E}^2\mathcal{C},\mathcal{F}}(t) \leq 2 \cdot \mathsf{Succ}^{\texttt{robust}}_{\mathcal{E}^2\mathcal{C}}(t) + 2 \cdot \mathsf{Succ}^{\texttt{s-bind-ext}}_{\mathcal{E}^2\mathcal{C}}(t) + 2 \cdot \mathsf{Adv}^{\texttt{smooth}}_{\mathcal{F}}$.

D.3 Proof of Theorem 7 (UC-Secure PAKE from an SPHF-Friendly Commitment)

To prove this theorem, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}_{\text{pwKE}}$ and the adversary $\mathcal{A}$.

For the sake of simplicity, since the protocol is fully symmetric in P_i and P_j, we describe the simulation for player P_i in order to simplify the notations.

We say that a flow is *oracle-generated* if the pair (hp, C) was sent by an honest player (or the simulator) and received without any alteration by the adversary. It is said *non-oracle-generated* otherwise.

Game $\boldsymbol{G}_0$: This is the **real game.**

Game $\boldsymbol{G}_1$: First, in this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs π_i and π_j sent by the environment to the players. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

In the following, Step 1. is always generated honestly by the simulator, since the hashing and projection keys do not depend on any private value.

Game $\boldsymbol{G}_2$: We now replace the setup algorithm $\mathsf{SetupCom}$ by $\mathsf{SetupComT}$ that additionally outputs the trapdoor $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under *setup indistinguishability.* Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: In the next two games, we deal with the case where P_i **receives a flow oracle-generated from P_j, and they have distinct passwords**. In this case, $\mathcal{S}$ has received the password π_j of P_j at the corruption time of P_j (π_j was anyway already known), and knows the corresponding opening data δ_j, generated with the commitment by the Com-call. If this password is the same, it does not change anything. If the passwords are distinct, then $\mathcal{S}$ computes H_i' as before, but chooses H_j at random: this means that we replace $\mathsf{Hash}(\mathsf{hk}_i, L_{\pi_i}, (\ell_j, C_j))$ by a random value, while C_j has been simulated by Com with an opening value δ_j for $\pi_j \neq \pi_i$.

With an hybrid proof, applying the $\mathsf{Exp}^{\texttt{c-smooth}}$ security game, with $x = \pi_i$ and $x' = \pi_j$ (since C_j is generated by Com on π_j, it thus extracts on $x' = \pi_j$), one proves this game is indistinguishable from the former one.

Game $\boldsymbol{G}_4$: We conclude for this case: if the passwords are distinct, P_i chooses a random key. Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_5$: In this game, we simulate the **commitments sent by the honest players** using the trapdoors generated by the setup algorithm $\mathsf{SetupComT}$. More precisely, we replace the commitment $(C_i, \delta_i) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell_i}(\pi_i)$ sent by an honest P_i with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$ in Step 2. by a simulated commitment $(C_i, \delta_i) \overset{\$}{\leftarrow} \mathsf{SCom}^{\ell_i}(\tau, \pi_i)$, which means $(C_i, \mathsf{eqk}_i) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell_i}(\tau)$ and $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}_i, C_i, \pi_i)$. We then store $(\ell_i, \pi_i, C_i, \delta_i)$ in Λ.

With an hybrid proof, applying the $\mathsf{Exp}^{\texttt{sim-ind}}$ security game for each player, in which SCom is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games.

In case of corruption of the honest player P_i, one learns the already known value π_i. If the corruption occurs before the erasures, we are able to provide the adversary with coherent values (δ_i has been computed using the correct value π_i). If the corruption occurs in the end, we are able to give the adversary the (honestly computed) session key. Unless we precise it, all the corruptions are dealt with in the same way in the following games.

Game $\boldsymbol{G}_6$: In this game, we deal with the case where P_i **receives a flow oracle-generated from P_j, and they have identical passwords.** When P_i receives an oracle-generated flow from P_j, the simulator checks whether the two passwords sent by the environment for P_i and P_j are identical. If so, $\mathcal{S}$ computes both hash values using Hash and not $\mathsf{ProjHash}$. More precisely, it computes $H_i' = \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_j}, (\ell_i, C_i))$ (with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$). If the passwords are distinct, it does not change anything. Recall that it is able to do so since it generated the hashing keys on their behalf.

Thanks to the correctness of the SPHF, this game is indistinguishable from the former one.

Game $\boldsymbol{G}_7$: Still in this case, we replace H_i' (and H_i if P_j received the oracle-generated flow generated flow sent by P_i) by a random value.

To prove this game is indistinguishable from the previous one, we consider two cases:

- P_j received the oracle-generated flow generated by P_i. In this case, hk_j is only used to compute $H_i = H'_i$, and since δ_i is no more used, we can apply the pseudo-randomness game on C_i to prove that $H_i = H'_i$ is indistinguishable from random;
- P_j received a non-oracle-generated flow (hp'_i, C'_i). In this case hk_j is only used to compute $H'_i = \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_i}, (\ell_i, C_i))$ and $H_i = \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_i}, (\ell_i, C'_i))$. In this case, we can apply the strong pseudo-randomness game to prove that H'_i still looks random.

Game $\boldsymbol{G}_8$: We conclude for this case: $\mathcal{S}$ sends a random key to P_i.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_9$: In the next two games, we deal with the case where P_i **receives a non-oracle-generated flow** (hp_j, C_j)**.** Since this pair is fresh, either C_j is new or hp_j (and thus the label) is new. In both cases, $\mathcal{S}$ can extract the committed value π'_j on behalf of P_j.

If this password is the same than that of P_i (which the simulator can easily check, still having access to the private values sent by the environment), $\mathcal{S}$ still computes both H_j and H'_i as before.

Otherwise (or if the extraction fails), the $\mathcal{S}$ computes H'_i as before, but chooses H_j at random: Under the smoothness, with an hybrid proof, applying the $\mathsf{Exp}^{\mathsf{c\text{-}smooth}}$ security game for each such hash value, one can show the indistinguishability of the two games.

Game $\boldsymbol{G}_{10}$: Finally, when P_i receives a non-oracle-generated flow (hp_j, C_j) that extracts to a different password than that of P_i (or for which extraction fails), then $\mathcal{S}$ sets the session key of P_i as random.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_{11}$: We do not use anymore the knowledge of π_i when simulating an honest player P_i. The simulator generates $(C_i, \mathsf{eqk}_i) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell_i}(\tau)$, with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$, to send C_i. It then stores $(\ell_i, \perp, C_i, \mathsf{eqk}_i)$ in Λ. We essentially break the atomic SCom in the two separated processes SimCom and $\mathsf{OpenCom}$. This does not change anything from the previous game since δ_i is never revealed. Λ is first filled with $(\ell_i, \perp, C_i, \mathsf{eqk}_i)$, it can be updated with correct values in case of corruption of P_i. Indeed, in case of corruption, $\mathcal{S}$ recovers the password π_i and computes $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}_i, C_i, \pi_i)$, which it is able to give to the adversary.

The private values of P_i are thus not used anymore in Step 1. and Step 2. The simulator only needs them to choose how to set the session key of the players. In the ideal game, this will be replaced by a `NewKey`-query that will automatically deal with equality or difference of the passwords, or `TestPwd`-query for non-oracle-generated-flows.

This game is perfectly indistinguishable from the former one.

Game $\boldsymbol{G}_{12}$: This is the ideal game. Now, the simulator does not know the private values of the honest players anymore, but can make use of the ideal functionality. We showed in Game $\boldsymbol{G}_{11}$ that the knowledge of the private values is not needed anymore by the simulator, provided it can ask queries to the ideal functionality:

Initialization: When initialized with security parameter $\mathfrak{K}$, the simulator first runs the commitment setup algorithm $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$, and initializes the real-world adversary $\mathcal{A}$, giving it ρ as common reference string.

Session Initialization: When receiving a message $(\texttt{NewSession}, \mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ from $\mathcal{F}_{pwKE}$, $\mathcal{S}$ executes the protocol on behalf of P_i as follows:

1. $\mathcal{S}$ generates honestly $\mathsf{hk}_i \overset{\$}{\leftarrow} \mathsf{HashKG}(L)$ and $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, L, \perp)$;
2. $\mathcal{S}$ computes $(C_i, \mathsf{eqk}_i) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell_i}(\tau)$ with $\ell_i = (\mathrm{sid}, P_i, P_j, \mathsf{hp}_i)$;
3. $\mathcal{S}$ sends $(\mathrm{sid}, P_i, P_j, \mathsf{hp}_i, C_i)$ to P_j.

If P_i gets corrupted, $\mathcal{S}$ recovers the password π_i and computes $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}, C_i, \pi_i)$, which it is able to give to the adversary.

Key Computation: When receiving a flow (hp_j, C_j):

– if the flow (C_j, hp_j) is non-oracle-generated, $\mathcal{S}$ extracts the password π'_j (or set it as a dummy value in case of failure of extaction). $\mathcal{S}$ then asked for a `TestPwd`-query to the functionality to check whether π'_j is the password of P_i. If this password is correct, $\mathcal{S}$ sets $\pi_i = \pi'_j$, computes $\delta_i \leftarrow \mathsf{OpenCom}^{\ell_i}(\mathsf{eqk}_i, C_i, \pi_i)$, as well as H_j and H'_i, and then sk_i, that is passed to the `NewKey`-query (`compromised` case). If the password is incorrect, $\mathcal{S}$ asks the `NewKey`-query with a random key (`interrupted` case).
– if the flow (C_j, hp_j) is oracle-generated but the associated P_j has been corrupted, then $\mathcal{S}$ has recovered its password π_j and δ_j. It can thus compute sk_j, that is passed to the `NewKey`-query (`corrupted` case).
– if the flow (C_j, hp_j) is oracle-generated and the associated P_j is still uncorrupted, $\mathcal{S}$ asks the `NewKey`-query with a random key (normal case).

One can remark that the `NewKey`-queries will send back the same kinds of session keys to the environment as in Game $\boldsymbol{G}_{11}$: if a player is corrupted, the really computed key is sent back, in case of impersonation attempt, the `TestPwd`-query will address the appropriate situation (correct or incorrect guess), and if the two players are honest, the `NewKey`-query also addresses the appropriate situation (same or different passwords).

More precisely, we have proven that for any environment, its advantage in distinguishing the ideal world (ideal functionality from Figure 10) and the real world (protocol from Figure 7) is bounded by $\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q \times \left(2 \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) + 3 \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{robust}}(t) + 2 \cdot \mathsf{Adv}_{\mathcal{F}}^{\texttt{smooth}} + \mathsf{Adv}_{\mathcal{C},\mathcal{F}}^{\texttt{c-s-ps-rand}}\right)$, in which q is the number of activated players and t its running time.

D.4 Proof of Theorem 8 (UC-Secure OT from an SPHF-Friendly Commitment)

To prove this theorem, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$. We prove the adaptive version of the protocol. The proof of the static version can be obtained by removing the parts related to adaptive version from the proof below.

Essentially, one first makes the setup algorithm additionally output the trapdoor (setup-indistinguishability); one can then replace all the commitment queries by simulated (fake) commitments (simulation-indistinguishability). When the sender submits the values $(\mathsf{hp}_i, M_i)_i$ the simulator can extract all the message thanks to the trapdoor and get the witnesses for each indices. This allows to simulate the `Send`-query to the ideal functionality. Eventually, when simulating the honest senders, the simulator extracts the committed value s, to set hp_s and M_s consistent with m_s, the other values can be random. More details follow:

Game $\boldsymbol{G}_0$: This is the real game.

Game $\boldsymbol{G}_1$: In this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs $(m_1, \dots, m_k)$ and s sent by the environment to the sender and the receiver. In all the subsequent games, the players use the label $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\boldsymbol{G}_2$: In this game, we just replace the setup algorithm $\mathsf{SetupCom}$ by $\mathsf{SetupComT}$ that additionally outputs the trapdoor $(\rho, \tau) \overset{\$}{\leftarrow} \mathsf{SetupComT}(1^{\mathfrak{K}})$, but nothing else changes, which does not alter much the view of the environment under *setup indistinguishability*. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: We first deal with **honest senders** P_i: when receiving a commitment C, the simulator extracts the committed value s. Instead of computing the key K_t, for $t = 1, \dots, k$ with the hash function, it chooses $K_t \overset{\$}{\leftarrow} G$ for $t \neq s$.

With an hybrid proof, applying the smoothness (see Figure 6 – left), for every honest sender, on every index $t \neq s$, since C is extracted to s, for any $t \neq s$, the hash value is indistinguishable from a random value.

In case of corruption, everything has been erased. This game is thus indistinguishable from the previous one under the smoothness.

Game $\boldsymbol{G}_4$: Still in this case, when receiving a commitment C, the simulator extracts the committed value s. Instead of proceeding as the sender would do on $(m_1, \dots, m_k)$, the simulator proceeds on $(m'_1, \dots, m'_k)$, with $m'_s = m_s$, but $m'_t = 0$ for all $t \neq s$. Since the masks K_t, for $t \neq s$, are random, this game is perfectly indistinguishable from the previous one.

Game $\boldsymbol{G}_5$: We now deal with **honest receivers** P_j: we replace all the commitments $(C, \delta) \overset{\$}{\leftarrow} \mathsf{Com}^\ell(s)$ with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ in Step 1 of the index query phase of honest receivers by simulated commitments $(C, \delta) \overset{\$}{\leftarrow} \mathsf{SCom}^\ell(\tau, s)$, which means $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^\ell(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^\ell(\mathsf{eqk}, C, s)$. We then store (ℓ, s, C, δ) in Λ.

With an hybrid proof, applying the $\mathsf{Exp}^{\texttt{s-sim-ind}}$ security game for each session, in which SCom is used as an atomic operation in which the simulator does not see the intermediate values, and in particular the equivocation key, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value s.

Game $\boldsymbol{G}_6$: We deal with **the generation of R for honest senders P_i on honestly-generated queries (adaptive case only)**: if P_i and P_j are honest at least until P_i received the second flow, the simulator sets $R = F(S')$ for both P_i and P_j, with S' a random value, instead of $R = F(S)$.

With an hybrid proof, applying the `IND-CPA` property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_7$: Still in the same case, the simulator sets R as a random value, instead of $R = F(S')$. With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_8$: We now deal with **the generation of K_s for honest senders P_i on honestly-generated queries**:

– in the static case (the pre-flow is not necessary, and thus we assume $R = 0$) the simulator chooses $K_s \overset{\$}{\leftarrow} G$ (for $t \neq s$, the simulator already chooses $K_t \overset{\$}{\leftarrow} G$), where s is the index given by the ideal functionality to the honest receiver P_j.
  With an hybrid proof, applying the pseudo-randomness (see Figure 6 – right), for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitments information δ for C;

– in the adaptive case, and thus with the additional random mask R, one can send a random M_s, and K_s can be computed later (when P_j actually receives its flow).
  As above, but only of P_j has not been corrupted before receiving its flow, the simulator chooses $K_s \overset{\$}{\leftarrow} G$. With an hybrid proof, applying the pseudo-randomness (see Figure 6 – right), for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitments information δ for C. If the player P_j involved in the pseudo-randomness game gets corrupted (but δ is unknown) we are not in this case, and we can thus abort it.
  In case of corruption of P_i, everything has been erased. In case of corruption of the receiver P_j, and thus receiving the value m_s, the simulator chooses R (because it was a random value unknown to the adversary and all the other K_t are independent random values too) such that

$$R \oplus \mathsf{ProjHash}(\mathsf{hp}_s, L_s, (\ell, C), \delta_s) \oplus M_s = m_s.$$

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Game G_9: Still in this case, the simulator proceeds on $(m'_1, \dots, m'_k)$, with $m'_t = 0$ for all i. Since the masks $K_t \oplus R$, for any $t = 1, \dots, k$, are independent random values (the K_t, for $t \neq s$ are independent random values, and K_s is also independently random in the static case, while R is independently random in the adaptive case), this game is perfectly indistinguishable from the previous one.
We remark that it is therefore no more necessary to know the index s given by the ideal functionality to the honest receiver P_j, to simulate P_i (but it is still necessary to simulate P_j).

Game G_{10}: We do not use anymore the knowledge of s when simulating an **honest receiver** P_j: the simulator generates $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$, to send C during the index query phase of honest receivers. It then stores $(\ell, \perp, C, \mathsf{eqk})$ in Λ. We essentially break the atomic SCom in the two separated processes SimCom and $\mathsf{OpenCom}$. This does not change anything from the previous game since δ is never revealed. Λ is first filled with $(\ell, \perp, C, \mathsf{eqk})$, it can be updated with correct values in case of corruption of the receiver.
When it thereafter receives $(\texttt{Send}, \mathrm{sid}, \mathrm{ssid}, P_i, P_j, (\mathsf{hp}_1, M_1, \dots, \mathsf{hp}_k, M_k))$ from the adversary, the simulator computes, for $i = 1, \dots, k$, $\delta_i \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, i)$, $K_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, (\ell, L_i), C, \delta_i)$ and $m_i = K_i \oplus R \oplus M_i$. This provides the database submitted by the sender.

Game G_{11}: We can now make use of the functionality, which leads to the following simulator:

- when receiving a `Send`-message from the ideal functionnality, which means that an honest sender has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted sender) and a `Receive`-message from the ideal functionality, which means that an honest receiver has sent an index query, the simulator generates $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$ and $c \overset{\$}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}, S)$, with $\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$ and R a random value, to send C and c during the index query phase of the honest receiver;
- when receiving a commitment C and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator extracts the committed value s, and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as S, and computes $R = F(S)$);
- when receiving $(\mathsf{hp}_1, M_1, \dots, \mathsf{hp}_k, M_k)$ from the adversary (a corrupted sender), the simulator computes, for $i = 1, \dots, k$, $\delta_i \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, i)$, $K_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, L_i, (\ell, C), \delta_i)$ and $m_i = K_i \oplus R \oplus M_i$. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with m_s, on behalf of a corrupted receiver, from the extracted s, instead of proceeding as the sender would do on $(m_1, \dots, m_k)$, the simulator proceeds on $(m'_1, \dots, m'_k)$, with $m'_s = m_s$, but $m'_i = 0$ for all $i \neq s$;
- when receiving a commitment C and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(m'_1, \dots, m'_k)$, with $m'_i = 0$ for all i, but it chooses R uniformly at random instead of choosing it as $R = F(S)$; in case of corruption afterward, the simulator will adapt R such that $R \oplus \mathsf{ProjHash}(\mathsf{hp}_s, L_s, (\ell, C), \delta_s) \oplus M_s = m_s$, where m_s is the message actually received by the receiver.

Any corruption either reveals s earlier, which allows a correct simulation of the receiver, or reveals $(m_1, \dots, m_k)$ earlier, which allows a correct simulation of the sender. When the sender has sent his flow, he has already erased all his random coins. However, there would have been an issue when the receiver is corrupted after the sender has sent is flow, but before the receiver receives it, since he has kept δ_s: this would enable the adversary to recover m_s from M_s and hp_s. This is the goal of the epheremal mask R that provides a secure channel.

As a consequence, the distance between the first and the last games is bounded by

$$\begin{aligned}
&\mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q_s \Big(\mathsf{Adv}_{\mathcal{E}}^{\texttt{ind-cpa}}(t) + \mathsf{Adv}_{F}^{\texttt{prg}}(t) + \mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) \\
&\qquad + (k-1)\mathsf{Adv}_{\mathcal{C},\mathcal{F}}^{\texttt{c-smooth}}(t) + \mathsf{Adv}_{\mathcal{C},\mathcal{F}}^{\texttt{c-ps-rand}}(t)\Big) \\
\leq\ & \mathsf{Adv}_{\mathcal{C}}^{\texttt{setup-ind}}(t) + q_s \times \Big(\mathsf{Adv}_{\mathcal{E}}^{\texttt{ind-cpa}}(t) + \mathsf{Adv}_{F}^{\texttt{prg}}(t) + 2 \cdot \mathsf{Adv}_{\mathcal{C}}^{\texttt{s-sim-ind}}(t) \\
&\qquad + k \cdot \big(\mathsf{Adv}_{\mathcal{C}}^{\texttt{robust}}(t) + \mathsf{Adv}_{\mathcal{F}}^{\texttt{smooth}}\big)\Big),
\end{aligned}$$

where q_s is the number of concurrent sessions and t the running time of the distinguisher.

E Concrete Instantiations

UC-Secure Password-Authenticated Key Exchange. A concrete instantiation of our generic PAKE construction from Figure 7 is presented in Figure 11.

$$\begin{array}{lcr}
 & \text{CRS}: \ (p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, h_1, f_1, c, d, T) & \\
P_i(\pi_i \in \{0,1\}^m) & \ell_i = (\text{sid}, P_i, P_j), \ell_j = (\text{sid}, P_j, P_i) & P_j(\pi_j \in \{0,1\}^m) \\
 & & \\
\mathsf{hk}_i = (\eta_{i,t,1}, \eta_{i,t,2}, \alpha_{i,t}, \beta_{i,t}, \mu_{i,t})_t \xleftarrow{\$} \mathbb{Z}_p^{5m} & & \mathsf{hk}_j = (\eta_{j,t,1}, \eta_{j,t,2}, \alpha_{j,t}, \beta_{j,t}, \mu_{j,t})_t \xleftarrow{\$} \mathbb{Z}_p^{5m} \\
\mathsf{hp}_i = (g_1^{\eta_{i,t,1}} h_1^{\alpha_{i,t}} f_1^{\beta_{i,t}} c^{\mu_{i,t}}, g_1^{\eta_{i,t,2}} d^{\mu_{i,t}})_t \in \mathbb{G}_1^{2m} & & \mathsf{hp}_j = (g_1^{\eta_{j,t,1}} h_1^{\alpha_{j,t}} f_1^{\beta_{j,t}} c^{\mu_{j,t}}, g_1^{\eta_{j,t,2}} d^{\mu_{j,t}})_t \in \mathbb{G}_1^{2m} \\
\pi_i = (\pi_{i,t})_t \in \{0,1\}^m; \ell'_i = (\ell_i, \mathsf{hp}_i) & & \pi_j = (\pi_{j,t})_t \in \{0,1\}^m; \ell'_j = (\ell_j, \mathsf{hp}_j) \\
(r_{i,t,\pi_{i,t}})_t \xleftarrow{\$} \mathbb{Z}_p^m; (r_{i,t,1-\pi_{i,t}})_t = (0)_t; \boldsymbol{s_i} \xleftarrow{\$} \mathbb{Z}_p^{2m} & & (r_{j,t,\pi_{j,t}})_t \xleftarrow{\$} \mathbb{Z}_p^m; (r_{j,t,1-\pi_{j,t}})_t = (0)_t; \boldsymbol{s_j} \xleftarrow{\$} \mathbb{Z}_p^{2m} \\
\boldsymbol{a_i} = (g_2^{r_{i,t,\pi_{i,t}}} T^{\pi_{i,t}})_t, \boldsymbol{d_i} = (g_1^{r_{i,t,u}})_{t,u} & & \boldsymbol{a_j} = (g_2^{r_{j,t,\pi_{j,t}}} T^{\pi_{j,t}})_t, \boldsymbol{d_j} = (g_1^{r_{j,t,u}})_{t,u} \\
\boldsymbol{b_i} = 2m\text{-}\mathsf{MCS}_{\mathsf{pk}}^{\ell'_i}(\boldsymbol{d_i}; \boldsymbol{s_i}) & & \boldsymbol{b_j} = 2m\text{-}\mathsf{MCS}_{\mathsf{pk}}^{\ell'_j}(\boldsymbol{d_j}; \boldsymbol{s_j}) \\
C_i = (\boldsymbol{a_i}, \boldsymbol{b_i}), \delta_i = (s_{i,t,\pi_{i,t}})_t & & C_j = (\boldsymbol{a_j}, \boldsymbol{b_j}), \delta_j = (s_{j,t,\pi_{j,t}})_t \\
\text{Erase everything, except } \delta_i & & \text{Erase everything, except } \delta_j \\
 & \xleftrightarrow[C_j, \mathsf{hp}_j]{C_i, \mathsf{hp}_i} & \\
H'_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, L_{\pi_i}, (\ell'_i, C_i), \delta_i) & & H'_j \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, L_{\pi_j}, (\ell'_j, C_j), \delta_j) \\
H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, L_{\pi_i}, (\ell'_j, C_j)) & & H_i \leftarrow \mathsf{Hash}(\mathsf{hk}_j, L_{\pi_j}, (\ell'_i, C_i)) \\
\mathsf{sk}_i \leftarrow H'_i \cdot H_j & & \mathsf{sk}_j \leftarrow H'_j \cdot H_i \\
\text{Erase everything, except } \pi_i, \mathsf{sk}_i & & \text{Erase everything, except } \pi_j, \mathsf{sk}_j
\end{array}$$

t ranges in $\{1, \ldots, m\}$, while u ranges in $\{0,1\}$.

Fig. 11. UC-Secure PAKE

UC-Secure Oblivious Transfert Protocol. A concrete instantiation of our generic OT constructions from Figure 8 is presented in Figure 12. No PRG is actually required here, and we can choose $S \xleftarrow{\$} \mathbb{G}_1$ and $R = e(S, g_2)$. The first flow and the ciphertext c needs not to be sent if only static security is required (in this case $R = 1$).

	CRS : $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, h_1, c, d, T)$	
$P_i(\boldsymbol{m} \in \mathbb{G}_T^k)$	$\ell = (\mathrm{sid}, \mathrm{ssid}, P_i, P_j)$	$P_j \begin{pmatrix} s \in \{1, \dots, k\} \\ s = (s_t)_t \in \{0,1\}^m \end{pmatrix}$
$\mathsf{sk} = s' \overset{\$}{\leftarrow} \mathbb{Z}_p; \mathsf{pk} = h' = g_1^{s'}$	$\xrightarrow{\mathsf{pk}}$	$(r_{t,s_t})_t \overset{\$}{\leftarrow} \mathbb{Z}_p^m; (r_{t,1-s_t})_t = (0)_t$
		$\boldsymbol{a} = (g_2^{r_{t,s_t}} T^{s_t})_t, \boldsymbol{d} = (g_1^{r_{t,u}})_{t,u}; \boldsymbol{z} \overset{\$}{\leftarrow} \mathbb{Z}_p^{2m}$
		$\boldsymbol{b} = 2m\text{-}\mathsf{MCS}^{\ell}_{\mathsf{pk}}(\boldsymbol{d}; \boldsymbol{z}); C = (\boldsymbol{a}, \boldsymbol{b}), \delta = (z_{t,s_t})_t$
		$S \overset{\$}{\leftarrow} \mathbb{G}_1; R = e(S, g_2); r' \overset{\$}{\leftarrow} \mathbb{Z}_p$
		$c = (c_1 = g^{r'}, c_2 = h'^{r'} \cdot S)$
$\varepsilon \overset{\$}{\leftarrow} \mathbb{Z}_p$	$\xleftarrow{C, c}$	Erase everything, except δ, R
$S = c_2 / c_1^{s'}; R = e(S, g_2)$		
for $v = 1$ to k :		
$\mathsf{hk}_v = (\eta_v, \alpha_v, \beta_v, \mu_v) \overset{\$}{\leftarrow} \mathbb{Z}_p^4$		
$\mathsf{hp}_v = (\varepsilon, g_1^{\eta_v} h_1^{\alpha_v} f_1^{\beta_v} (cd^{\theta})^{\mu_v}) \in \mathbb{Z}_p \times \mathbb{G}_1$		
$M_v \leftarrow R \cdot m_v \cdot \mathsf{Hash}((\varepsilon, \mathsf{hk}_v), v, (\ell, C))$		
Erase everything, except $\boldsymbol{M}, \mathbf{hp}$	$\xrightarrow{\boldsymbol{M}, \mathbf{hp}}$	$m_s \leftarrow M_s / (R \cdot \mathsf{ProjHash}(\mathsf{hp}_s, s, (\ell, C), \delta))$
		Erase everything, except s, m_s

v ranges in $\{1, \dots, k\}$, while t ranges in $\{1, \dots, m\}$, for $m = \lceil \log k \rceil$, and u ranges in $\{0, 1\}$.

Fig. 12. UC-Secure 1-out-of-k OT Protocol

Appendix D

Adaptive Oblivious Transfer and Generalization [BCG16]

This is the Full Version of the Extended Abstract that appears in Advances in Cryptology — Proceedings of ASIACRYPT'2016 (4 – 8 December 2016, Hanoi, Vietnam), Jung Hee Cheon and Tsuyoshi Takagi, Springer-Verlag, Part II, LNCS 10032, pages 217–247.

Authors

Olivier Blazy, Céline Chevalier, Paul Germouty

Abstract

Oblivious Transfer (OT) protocols were introduced in the seminal paper of Rabin, and allow a user to retrieve a given number of lines (usually one) in a database, without revealing which ones to the server. The server is ensured that only this given number of lines can be accessed per interaction, and so the others are protected; while the user is ensured that the server does not learn the numbers of the lines required. This primitive has a huge interest in practice, for example in secure multi-party computation, and directly echoes to Symmetrically Private Information Retrieval (SPIR).

Recent Oblivious Transfer instantiations secure in the UC framework suffer from a drastic fallback. After the first query, there is no improvement on the global scheme complexity and so subsequent queries each have a global complexity of $\mathcal{O}(|\mathrm{DB}|)$ meaning that there is no gain compared to running completely independent queries. In this paper, we propose a new protocol solving this issue, and allowing to have subsequent queries with a complexity of $\mathcal{O}(\log(|\mathrm{DB}|))$ while keeping round optimality, and prove the protocol security in the UC framework with adaptive corruptions and reliable erasures.

As a second contribution, we show that the techniques we use for Oblivious Transfer can be generalized to a new framework we call *Oblivious Language-Based Envelope* (OLBE). It is of practical interest since it seems more and more unrealistic to consider a database with uncontrolled access in access control scenarios. Our approach generalizes Oblivious Signature-Based Envelope, to handle more expressive credentials and requests from the user. Naturally, OLBE encompasses both OT and OSBE, but it also allows to achieve Oblivious Transfer with fine grain access over each line. For example, a user can access a line if and only if he possesses a certificate granting him access to such line.

We show how to generically and efficiently instantiate such primitive, and prove them secure in the Universal Composability framework, with adaptive corruptions assuming reliable erasures. We provide the new UC ideal functionalities when needed, or we show that the existing ones fit in our new framework.

The security of such designs allows to preserve both the secrecy of the database values and the user credentials. This symmetry allows to view our new approach as a generalization of the notion of Symmetric PIR.

1 Introduction

Oblivious Transfer (OT) is a notion introduced by Rabin in [Rab81]. In its classical 1-out-of-n version, it allows a user $\mathcal{U}$ to access a single line of a database while interacting with the server $\mathcal{S}$ owning the database. The user should be oblivious to the other line values, while the server should be oblivious to which line was indeed received. Oblivious transfer has a fundamental role for achieving secure multi-party computation: It is for example needed for every bit of input in Yao's protocol [Yao86] as well as for Oblivious RAM ([WHC+14] for instance), for every AND gate in the Boolean circuit computing the function in [GMW87] or for almost all known garbled circuits [BHR12].

Private Information Retrieval (PIR) schemes [CGKS95] allow a user to retrieve information from a database, while ensuring that the database does not learn which data were retrieved. With the increasing need for user privacy, these schemes are quite useful in practice, be they used for accessing records for email repositories, collection of webpages, music... But while protecting the privacy of the user, it is equally important that the user should not learn more information than he is allowed to. This is called database privacy and the corresponding protocol is called a Symmetrically Private Information Retrieval (SPIR), which could be employed in practice, for medical data or biometric information. This notion is closely related to Oblivious Transfer.

Due to their huge interest in practice, it is important to achieve low communication on these Oblivious Transfer protocols. A usual drawback is that the server usually has to send a message equivalent to the whole database each time the user requests a line. If it is logical, in the UC framework, that an OT protocol requires a cost linear in the size of the database for the first line queried. One may then hope to amortize the cost for further queries between the same server and the same user (or even another user, if possible), reducing the efficiency gap between Private Information Retrieval schemes and their stronger equivalent Oblivious Transfer schemes. We thus deal in this paper with a more efficient way, which is to achieve *Adaptive* Oblivious Transfer, in which the user can adaptively ask several lines of the database. In such schemes, the server only sends his database once at the beginning of the protocol, and all the subsequent communication is in $o(n)$, more precisely logarithmic. The linear cost is batched once and for all in this preprocessing phase, achieving then a logarithmic complexity similar to the best PIR schemes.

Smooth Projective Hash Functions (SPHF), used in conjunction with *Commitments* have become the standard way to deal with such secret message transfers. In a commitment scheme, the sender is going to commit to the line required (*i.e.* to give the receiver an analogue of a sealed envelope containing his value i) in such a way that he should not be able to open to a value different from the one he committed to (*binding* property), and that the receiver cannot learn anything about i (*hiding* property) before a potential opening phase. During the opening phase, however, the committer would be asked to reveal i in such a way that the receiver can verify it was indeed i that was contained in the envelope.

But, in our applications, there cannot be an opening phase, due to the *oblivious* requirements on the protocols and the secrecy of the database line i sent. The decommitment (opening phase) will thus be implicit, which means that the committer does not really open its commitment, but rather convinces the receiver that it actually committed to the value it pretended to. We achieve this property thanks to *Smooth Projective Hash Functions* [CS02, GL03], which have been widely used in such circumstances (see [ACP09, KV11, BBC+13b, ABB+13, BC15] for instance). These hash functions are defined in such a way that their value can be computed in two different ways if the input belongs to a particular subset (the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language. The hash value obtained is indistinguishable from random in case the input does not belong to the language (*smoothness*) and in case the input does belong to the language but no witness is known (*pseudo-randomness*).

In a nutshell, to ensure implicit decommitment, the sender will thus simply mask the database line with this hash value computed using the private hashing key. He will then send it along with the public projection key to the user, who will be able to compute the same hash value thanks to the randomness of the commitment of this line he sent in the first place (the randomness is the witness of the membership of the commitment to the language of commitments of this specific line). In order to ensure adaptive security in the universal composability framework, the commitments used are usually required to be both *extractable* (meaning that a simulator can recover the value i committed to thanks to a trapdoor) and *equivocable* (meaning that a simulator can open a commitment to a value i' different from the value i it committed to thanks to a trapdoor).

In order to simplify these commitments, which can be quite technical, we choose here to rely on words in more complex languages rather than on simple line numbers. More precisely, the user will first compute an equivocable commitment on the line number required, which will be his word w in the language. This word will then be encrypted under a CCA encryption scheme, and the SPHF will be constructed for this word (rather than for the line number), which will be simpler. Furthermore, this abstraction consisting in encoding line numbers as words in more complex languages will reveal useful in more general contexts, not only Oblivious Transfer, the simplest of which being Oblivious Signature Based Envelope.

Oblivious Signature-Based Envelope (OSBE) was introduced by Li, Du and Boneh in [LDB03]. OSBE schemes consider the case where Alice (the receiver) is a member of an organization and possesses a certificate produced by an authority attesting she actually belongs to this organization. Bob (the sender) wants to send a private message P to members of this organization. However due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization. OSBE lets Bob send an obfuscated version of this message P to Alice, in such a way that she will be able to find P if and only if she is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization. We even manage to construct a more general framework to capture many protocols around trust negotiation, where the user receives a message if and only if he possesses some credentials or specific accreditations. As a reference to OSBE, we call this framework *Oblivious Language-Based Envelope* (OLBE).

1.1 Related Work

Since the original paper [Rab81], several instantiations and optimizations of OT protocols have appeared in the literature [NP01, CLOS02], including proposals in the UC framework. More recently, new instantiations have been proposed, trying to reach round-optimality [HK07], and/or low communication costs [PVW08]. Recent schemes like [ABB+13, BC15] manage to achieve round-optimality while maintaining a small communication cost. Choi *et al.* [CKWZ13] also propose a generic method and an efficient instantiation secure against adaptive corruptions in the CRS model with erasures, but it is only 1-out-of-2 and it does not scale to 1-out-of-n OT, for $n > 2$. As far as adaptive versions of those protocols are concerned, this problem was first studied by [NP97, GH07, KNP11], and more recently UC secure instantiations were proposed, but unfortunately either under the Random Oracle, or under not so standard assumptions such as q-Hidden LRSW or later on q-SDH [CNs07, JL09, RKP09, CDH12, GD14], but without allowing adaptive corruptions.

Concerning automated trust negotiation, two frameworks have been proposed to encompass the symmetric protocols (Password-based Authenticated Key-Exchange, Secret Handshakes and Verfier-Based PAKE): The Credential Authenticated Key Exchange [CCGS10], and Language-based Authenticated Key Exchange (LAKE) [BBC+13a], in which two parties establish a common session key if and only if they hold credentials that belong to specific (and possibly independent) languages chosen by the other party. As for OSBE, the authors in [BPV12] improved the security model initially proposed in [LDB03], showing how to use Smooth Projective Hash Functions to do implicit proof of knowledge, and proposed the first efficient instantiation of OSBE, under a standard hypothesis. It fits, as well as Access Controlled Oblivious Transfer [CDN09, CDNZ11], Priced Oblivious Transfer [AIR01, RKP09]) and Conditional Oblivious Transfer [DOR99], into the generic notion of Conditional Disclosure of Secrets (see for instance [AIR01, GIKM98, BGN05, LL07, Wee14, IW14, Att14, GKW15]).

1.2 Contributions

Our first contribution is to give the first round-optimal adaptive Oblivious Transfer protocol secure in the UC framework with adaptive corruptions under standard assumptions (MDDH) and assuming reliable erasures. We show how to instantiate the needed building blocks using standard assumptions, using or extending various basic primitives in order to fit the MDDH framework introduced in [EHK$^+$13]. In our scheme, the server first preprocesses its database in a time linear in the length of the database and transfers it to the receiver. After that, the receiver and the sender can run many instances of the protocol on the same database as input and adaptively chosen inputs from the receiver, with a cost sublinear in the database.

It is interesting to note that our resulting adaptive Oblivious Transfer scheme has an amortized complexity in $\mathcal{O}(\log |DB|)$, which is similar to current Private Information Retrieval instantiations [KLL$^+$15], that have weaker security prerequisites, and much better than current UC secure Oblivious Transfer under standard assumptions (as they are in $\mathcal{O}(|DB|)$. For a fair comparison it should be stated that the PIR schemes allow this complexity directly from the first query while in our case due to the preprocessing, this amortized cost is only reached after a high number of queries. However, it is interesting to see this convergence in spite of hugely different security models and expectation. Compared to existing versions cited above (either proven in classical security models, or in the UC framework but only with static corruptions and under non-standard assumptions), we manage to prove its security under standard assumptions, like SXDH, and allow UC security with adaptive user corruptions.

As a side result, it is worth noting that we follow some ideas developed in the construction explained in [GH07] around Blind Identity-Based Encryption and provide techniques in order to transform IBE schemes into blind ones, applying them to revisit the one given in [BKP14], in order to show how we can answer blind user secret key-retrieval, which can be of independent interest.

As a second contribution, we propose our new notion, that we call Oblivious Language-Based Envelope. We provide a security model by giving a UC ideal functionality, and show that this notion supersedes the classical asymmetric automated trust negotiation schemes recalled above such as Oblivious Transfer and Oblivious Signature-Based Envelope. We show how to choose the languages in order to obtain from our framework all the corresponding ideal functionalities, recovering the known ones (such as OT) and providing the new ones (such as OSBE, to the best of our knowledge). We then give a generic construction scheme fulfilling our ideal functionality, which directly gives generic constructions for the specific cases (OT, OSBE). Finally, we show how to instantiate the different simple building blocks in order to recover the standard efficient instantiations of these schemes from our framework. In addition to the two cases most studied (OT, OSBE), we also propose what we call *Conditioned Oblivious Transfer*, which encompasses Access Controlled Oblivious Transfer, Priced Oblivious Transfer and Conditional Oblivious Transfer, and in which the access to each line of the database is hidden behind some possibly secret restriction, be it a credential, a price, or an access policy. The advantage of the OLBE framework on the notion of Conditional Disclosure of Secrets is to allow generic constructions of a large subclass of schemes, as long as two participants are involved. It can be easily applied to any language expressing some new access control policy. Furthermore, those instantiations fit into a global security model, allowing to uniformize (for the better) the security expectations for such schemes. In particular, we allow security in the UC framework with adaptive corruptions for all our constructions (which was already known for some primitives cited above, but not all), and manage to achieve this level of security while staying in the standard model with standard hypothesis.

2 Definitions and Building Blocks

2.1 Notations for Classical Primitives

Throughout this paper, we use the notation $\mathfrak{K}$ for the security parameter.

Digital Signature. A digital signature scheme $\mathcal{S}$ [DH76, GMR88] allows a signer to produce a verifiable proof that he indeed produced a message. It is described through four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. The formal definitions are given in Appendix A.

Encryption. An encryption scheme $\mathcal{C}$ is described by four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$. The formal definitions are given in Appendix A.

Commitment and Chameleon Hash. Commitments allow a user to commit to a value without revealing it, but without the possibility to later change his mind. It is composed of four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Commit}, \mathsf{Decommit})$. Informally, it is extractable if a simulator knowing a certain trapdoor can recover the value committed to, and it is equivocable if a simulator, knowing another trapdoor, can open the commitment to another value than the one it actually committed to. This directly echoes to Chameleon Hashes, traditionally defined by three algorithms $\mathsf{CH} = (\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll})$. The formal definitions are given in Appendix A.

2.2 Identity-Based Encryption, Identity-based Key Encapsulation

Identity Based encryption was first introduced by Shamir in [Sha84] who was expecting an encryption scheme where no public key will be needed for sending a message to a precise user, defined by his identity. Thus any user wanting to send a private message to a user only need this user's identity and a master public key. It took 17 years for the cryptographic community to find a way to realize this idea. The first instantiation was proposed in [BF01] by Boneh and Franklin. It can be described as an identity-based key encapsulation (IBKEM) scheme IBKEM which consists of four algorithms $\mathsf{IBKEM} = (\mathsf{Gen}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$. Every IBKEM can be transformed into an ID-based encryption scheme IBE using a (one-time secure) symmetric cipher.

Definition 1 (Identity-based Key Encapsulation Scheme).
An identity-based key encapsulation scheme IBKEM *consists of four PPT algorithms* $\mathsf{IBKEM} = (\mathsf{Gen}, \mathsf{USKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties.*

- $\mathsf{Gen}(\mathfrak{K})$*: returns the (master) public/secret key* $(\mathsf{mpk}, \mathsf{msk})$. *We assume that* mpk *implicitly defines an identity space* $\mathcal{ID}$*, a key space* $\mathcal{KS}$*, and ciphertext space* CS.
- $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$*: returns the user secret-key* $\mathsf{usk}[\mathsf{id}]$ *for identity* $\mathsf{id} \in \mathcal{ID}$.
- $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$*: returns the symmetric key* $\mathsf{K} \in \mathcal{KS}$ *together with a ciphertext* $\mathsf{C} \in \mathsf{CS}$ *with respect to identity* id.
- $\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$*: returns the decapsulated key* $\mathsf{K} \in \mathcal{KS}$ *or the reject symbol* $\perp$.

For perfect correctness we require that for all $\mathfrak{K} \in \mathbb{N}$*, all pairs* $(\mathsf{mpk}, \mathsf{msk})$ *generated by* $\mathsf{Gen}(\mathfrak{K})$*, all identities* $\mathsf{id} \in \mathcal{ID}$*, all* $\mathsf{usk}[\mathsf{id}]$ *generated by* $\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$ *and all* (K, C) *output by* $\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$*:* $\Pr[\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C}) = \mathsf{K}] = 1$.

The security requirements for an IBKEM we consider here are indistinguishability and anonymity against chosen plaintext and identity attacks (IND-ID-CPA and ANON-ID-CPA). Instead of defining both security notions separately, we define pseudorandom ciphertexts against chosen plaintext and identity attacks (PR-ID-CPA) which means that challenge key and ciphertext are both pseudorandom. Note that PR-ID-CPA trivially implies IND-ID-CPA and ANON-ID-CPA. We define PR-ID-CPA-security of IBKEM formally via the games given in Figure 1.

```
Procedure Initialize:
    $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Gen}(\mathfrak{K})$
    Return $\mathsf{mpk}$

Procedure USKGen(id):
    $\mathcal{Q}_{\mathcal{ID}} \leftarrow \mathcal{Q}_{\mathcal{ID}} \cup \{\mathsf{id}\}$
    Return $\mathsf{usk}[\mathsf{id}] \xleftarrow{\$} \mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$

Procedure Enc(id*):
    // one query
    $(\mathsf{K}^*, \mathsf{C}^*) \xleftarrow{\$} \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*)$
    $\boxed{\mathsf{K}^* \xleftarrow{\$} \mathcal{KS}; \mathsf{C}^* \xleftarrow{\$} \mathsf{CS}}$
    Return $(\mathsf{K}^*, \mathsf{C}^*)$

Procedure Finalize(β):
    Return $(\mathsf{id}^* \notin \mathcal{Q}_{\mathcal{ID}}) \wedge \beta$
```

Fig. 1. PR-ID-CPA-security: Security Games $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{real}}$ and $\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{rand}}$ (boxed).

Definition 2 (PR-ID-CPA Security). *An ID-based key encapsulation scheme* IBKEM *is said* PR-ID-CPA-*secure if for all PPT $\mathscr{A}$, the following advantage is negligible:* $\mathsf{Adv}^{\mathsf{pr\text{-}id\text{-}cpa}}_{\mathsf{IBKEM}}(\mathscr{A}) := |\Pr[\mathsf{PR\text{-}ID\text{-}CPA}^{\mathscr{A}}_{\mathsf{real}} \Rightarrow 1] - \Pr[\mathsf{PR\text{-}ID\text{-}CPA}_{\mathsf{rand}}{}^{\mathscr{A}} \Rightarrow 1]|$.

2.3 Smooth Projective Hashing and Languages

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup in [CS02] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found applications in various contexts in cryptography (*e.g.* [GL03, Kal05, ACP09]). A Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$, onto a set $\mathcal{G}$, is defined by five algorithms ($\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash}$):

– $\mathsf{Setup}(1^{\mathfrak{K}})$ where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme, and the description of an $\mathcal{NP}$ language $\mathfrak{L}$;
– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key hk for the language $\mathfrak{L}$;
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, derives the projection key hp from the hashing key hk and the word W.
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs a hash value $v \in \mathcal{G}$, using the hashing key hk and the word W.
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, outputs the hash value $v' \in \mathcal{G}$, using the projection key hp and the witness w that the word $W \in \mathfrak{L}$.

In the following, we assume $\mathfrak{L}$ is a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $X \setminus \mathfrak{L}$. An SPHF should satisfy the following properties:

– *Correctness*: Let $W \in \mathfrak{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have
$$\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w).$$
– *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathcal{G} \end{array} \right\}.$$

This is formalized by: $\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{SPHF}}(\mathfrak{K}) = \sum_{V \in \mathbb{G}} |\Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V]|$ is negligible.

– *Pseudo-Randomness*: If $W \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable. For any adversary $\mathcal{A}$ within reasonable time, this advantage is negligible:

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]|$$

Languages. The language $\mathfrak{L} \subset X$ used in the definition of an SPHF should be a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element not in $\mathfrak{L}$ (see formal definition in [GL03, AP06]). The languages used here are more complex and should fulfill the following properties[1]:

– *Publicly Verifiable:* Given a word x in X, anyone should be able to decide in polynomial time whether $x \in \mathfrak{L}$ or not.
– *Self-Randomizable*: Given a word in the language, anyone should be able to sample a new word in the language[2], and the distribution of this resampling should be indistinguishable from an honest distribution. This will be used in order to prevent an adversary, or the authority in charge of distributing the words, to learn which specific form of the word was used by the user.

In case we consider several languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$, we also assume it is a *Trapdoor Collection of Languages*: It is computationally hard to sample an element in $\mathfrak{L}_1 \cap \dots \cap \mathfrak{L}_n$, except if one possesses a trapdoor tk (without the knowledge of the potential secret keys)[3]. For instance, if for all i, $\mathfrak{L}_i$ is the language of the equivocable commitments on words in an inner language $\widetilde{\mathfrak{L}_i} = \{i\}$ (as we will consider for OT), the common trapdoor key can be the equivocation trapdoor.

Depending on the applications, we can assume a *Keyed Language*, which means that it is set by a trusted authority, and that it is hard to sample fresh elements from scratch in the language without the knowledge of a secret language key $\mathsf{sk}_{\mathfrak{L}}$. In this case, the authority is also in charge of giving a word in the language to the receiver.

In case the language is keyed, we assume it is also a *Trapdoor Language*: We assume the existence of a trapdoor tk_L allowing a simulator to sample an element in $\mathfrak{L}$ (without the knowledge of the potential secret key $\mathsf{sk}_{\mathfrak{L}}$). For instance, for a language of valid Waters signatures of a message M (as we will consider for OSBE), one can think of $\mathsf{sk}_{\mathfrak{L}}$ as being the signing key, whereas the trapdoor $\mathsf{tk}_{\mathfrak{L}}$ can be the discrete logarithm of h in basis g.[4]

[1] We here mainly consider languages which are hard-partitioned subsets, for instance, encryptions of publicly verifiable languages.
[2] It should be noted that this property is not incompatible with the potential secret key of the language in case it is keyed (see below).
[3] This implicitly means that the languages are compatible, in the sense that one can indeed find a word belonging to all of them.
[4] As another example, one may think of more expressive languages which may not rely directly on generators fixed by the CRS. In this case, one can assume that the CRS contains parameters for an encryption and an associated NIZK proof system. The description of such a language is thus supplemented with an encryption of the language trapdoor, and a non-interactive zero-knowledge proof that the encrypted value is indeed a trapdoor for the said language. Using the knowledge of the decryption key, the simulator is able to recover the trapdoor.

2.4 Security Assumptions

Due to lack of space, instantiations of the primitives recalled above are given in Appendix B and we only give here the security assumptions.

Security Assumption: Pairing groups and Matrix Diffie-Hellman Assumption. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^{\mathfrak{K}}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

We use implicit representation of group elements as introduced in [EHK$^+$13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$. More generally, for a matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ we define $[\boldsymbol{A}]_s$ as the implicit representation of $\boldsymbol{A}$ in $\mathbb{G}_s$:

$$[\boldsymbol{A}]_s := \begin{pmatrix} g_s^{a_{11}} \dots g_s^{a_{1m}} \\ \\ g_s^{a_{n1}} \dots g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [b]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^k$ define $e([\boldsymbol{a}]_1, [\boldsymbol{b}]_2) := [\boldsymbol{a}^\top \boldsymbol{b}]_T \in \mathbb{G}_T$. We recall the definition of the matrix Diffie-Hellman (MDDH) assumption [EHK$^+$13].

Definition 3 (Matrix Distribution). *Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.*

Without loss of generality, we assume the first k rows of $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k$ form an invertible matrix, we denote this matrix $\overline{\boldsymbol{A}}$, while the last line is denoted $\underline{A}$. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\boldsymbol{A}], [\boldsymbol{A}\boldsymbol{w}])$ and $([\boldsymbol{A}], [\boldsymbol{u}])$ where $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k$, $\boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

Definition 4 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH). *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption holds relative to GGen in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) := |\Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{A}\boldsymbol{w}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{u}]_s) = 1]| = \mathsf{negl}(\lambda),$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

For each $k \geq 1$, [EHK$^+$13] specifies distributions $\mathcal{L}_k$, $\mathcal{U}_k$, ... such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-uniform and others. All assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. The distributions are exemplified for $k = 2$, where $a_1, \dots, a_6 \xleftarrow{\$} \mathbb{Z}_p$.

$$\mathcal{L}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{U}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}.$$

It was also shown in [EHK$^+$13] that $\mathcal{U}_k$-MDDH is implied by all other $\mathcal{D}_k$-MDDH assumptions.

Lemma 5 (Random self reducibility [EHK+13]). *For any matrix distribution $\mathcal{D}_k$, $\mathcal{D}_k$-*MDDH *is random self-reducible. In particular, for any $m \geq 1$ and for all PPT adversaries $\mathcal{D}$ and $\mathcal{D}'$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) + \tfrac{1}{q-1} \geq \mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}')$$

where $\mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}') := \Pr[\mathcal{D}'(\mathcal{G},[\boldsymbol{A}],[\boldsymbol{AW}]) \Rightarrow 1] - \Pr[\mathcal{D}'(\mathcal{G},[\boldsymbol{A}],[\boldsymbol{U}]) \Rightarrow 1]$, *with* $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \overset{\$}{\leftarrow} \mathcal{D}_k, \boldsymbol{W} \overset{\$}{\leftarrow} \mathbb{Z}_p^{k\times m}, \boldsymbol{U} \overset{\$}{\leftarrow} \mathbb{Z}_p^{(k+1)\times m}$.

Remark: It should be noted that $\mathcal{L}_1, \mathcal{L}_2$ are respectively the SXDH and DLin assumptions.

2.5 Security Models

UC Framework. The goal of the UC framework [Can01] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol $\mathit{\Pi}$ emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathscr{A}$ (which controls the communication between the players), a simulator $\mathscr{S}$ such that no polynomial environment $\mathscr{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathscr{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathscr{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathscr{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

Simple UC Framework. Canetti, Cohen and Lindell formalized a simpler variant in [CCL15], that we use here. This simplifies the description of the functionalities for the following reasons (in a nutshell): All channels are automatically assumed to be authenticated (as if we worked in the $\mathcal{F}_{\text{AUTH}}$-hybrid model); There is no need for *public delayed outputs* (waiting for the adversary before delivering a message to a party), neither for an explicit description of the corruptions. We refer the interested reader to [CCL15] for details.

3 UC-secure Adaptive Oblivious Transfer

As explained in the introduction, the classical OT constructions based on the commitment/SPHF paradigm (with so-called implicit decommitment), among the latest schemes in the UC framework [CKWZ13, ABB+13, BC15], require the server to send an encryption of the complete database for each line required by the user (thus $O(n)$ each time). We here give a protocol requiring $O(\log(n))$ for each line (except the first one, still in $O(n)$), in the UC framework with adaptive corruptions under classical assumptions (MDDH). This protocol builds upon the more efficient known scheme secure in the UC framework [BC15] and we use ideas from [GH07] to make it adaptive.

Using implicit decommitment in the UC framework implies a very strong commitment primitive (formalized as SPHF-friendly commitments in [ABB+13]), which is both extractable and equivocable. Our idea is here to split these two properties by using on the one hand an equivocable commitment and on the other hand an (extractable) CCA encryption scheme by generalizing the way to access a line in the database. But this is infeasible with simple line numbers. Indeed, we suggest here not to consider anymore the line numbers as numbers in $\{1,\ldots,n\}$ but rather to "encode" them (the exact encoding will depend on the protocol): For every line i, a word W_i in the language $\mathfrak{L}_i$ will correspond to a representation of line i. This representation must be publicly verifiable, in the sense that anyone can associate i to a word W_i. We formalize this in the following definition of oblivious transfer[5], given without loss of generality[6] (the classical notion of OT being easily captured using $\mathfrak{L}_i = \{i\}$).

3.1 Definition and Security Model for Oblivious Transfer

In such a protocol, a server $\mathcal{S}$ possesses a database of n lines $(m_1, \dots, m_n) \in (\{0,1\}^{\mathfrak{K}})^n$. A user $\mathcal{U}$ will be able to recover m_k (in an oblivious way) as soon as he owns a word $W_k \in \mathfrak{L}_k$. The languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$ will be assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Definition 6 (Oblivious Transfer). *An* OT *scheme is defined through five algorithms* (Setup, KeyGen, DBGen, Samp, Verify)*, along with an interactive protocol* $\mathsf{Protocol}\langle\mathcal{S},\mathcal{U}\rangle$*:*

– $\mathsf{Setup}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the number* n*;*

or $\mathsf{SetupT}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[7] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$*.*

– $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ *generates, for all* $i \in \{1, \dots, n\}$*, the description of the language* $\mathfrak{L}_i$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_i}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$*.*

– $\mathsf{Samp}(\mathsf{param})$ *or* $\mathsf{Samp}(\mathsf{param}, (\mathsf{sk}_{\mathfrak{L}_i})_{i\in\{1,\dots,n\}})$ *generates a word* $W_i \in \mathfrak{L}_i$*;*

– $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *checks whether* W_i *is a valid word in the language* $\mathfrak{L}_i$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*

– $\mathsf{Protocol}\langle\mathcal{S}((\mathfrak{L}_1, \dots, \mathfrak{L}_n), (m_1, \dots, m_n)), \mathcal{U}((\mathfrak{L}_1, \dots, \mathfrak{L}_n), W_i)\rangle$*, which is executed between the server* $\mathcal{S}$ *with the private database* $(m_1, \dots, m_n)$ *and corresponding languages* $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$*, and the user* $\mathcal{U}$ *with the same languages and the word* W_i*, proceeds as follows. If the algorithm* $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *returns* 1*, then* $\mathcal{U}$ *receives* m_k*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

The ideal functionality of an Oblivious Transfer (OT) protocol was given in [Can01,CKWZ13, ABB$^+$13], and an adaptive version in [GH08]. We here combine them and rewrite it in simple UC and using our language formalism (instead of directly giving a number line s to the functionality, the user will give it a word $W_s \in \mathfrak{L}_s$). The resulting functionality $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$ is given in Figure 2. Recall that there is no need to give an explicit description of the corruptions in the simple version of UC [CCL15].

3.2 High Level Idea of the Construction of the Adaptive Oblivious Transfer Scheme

Our construction builds upon the UC-secure OT scheme from [BC15], with ideas inspired from [GH07], who propose a neat framework allowing to achieve *adaptive* Oblivious Transfer (but not in the UC framework). Their construction is quite simple: It requires a *blind Identity-Based Encryption*, in other words, an IBE scheme in which there is a way to query for a user key generation without the authority (here the server) learning the targeted identity (here the line in the database). Once such a Blind IBE is defined, one can conveniently obtain an oblivious transfer

[5] The adaptive version only implies that the database $(m_1, \dots, m_n)$ is sent only once in the interaction, while the user can query several lines (*i.e.* several words), in an adaptive way.

[6] This formalization furthermore encompasses the variants of OT, such as conditioned OT, where a user accesses a line only if he knows a credential for this line.

[7] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

The functionality $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1,\ldots,\mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1,\ldots,\mathfrak{P}_N$ via the following queries:

- **Upon receiving from party $\mathfrak{P}_i$ an input $(\mathtt{NewDataBase}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1,\ldots,m_n))$** , with $m_k \in \{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(m_1,\ldots,m_n))$ and reveal $(\mathtt{Send},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further `NewDataBase`-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, W_k)$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(m_1,\ldots,m_n))$ is not recorded. Otherwise, reveal $(\mathtt{Receive},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send the message $(\mathtt{Received},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,m'_k)$ to $\mathfrak{P}_j$ where $m'_k = m_k$ if $\mathsf{Verify}_k(W_k,\mathfrak{L}_k)$ returns 1, and $m'_k = \perp$ otherwise.

(Non-Adaptive case: Ignore further `Receive`-message with the same ssid *from* $\mathfrak{P}_j$*.)*

Fig. 2. Ideal Functionality for (Adaptive) Oblivious Transfer $\mathcal{F}_{\mathrm{OT}}^{\mathfrak{L}}$

protocol by asking the database to encrypt (once and for all) each line for an identity (the j-th line being encrypted for the identity j), and having the user do a blind user key generation query for identity i in order to recover the key corresponding to the line i he expects to learn.

This approach is round-optimal: After the database preparation, the first flow is sent by the user as a commitment to the identity i, and the second one is sent by the server with the blinded expected information. But several technicalities arise because of the UC framework we consider here. For instance, the blinded expected information has to be masked, we do this here thanks to an SPHF. Furthermore, instead of using simple line numbers as identities, we have to commit to words in specific languages (so as to ensure extractability and equivocability) as well as to *fragment* the IBE keys into bits in order to achieve $O(\log n)$ in both flows. This allows us to achieve the first UC-secure adaptive OT protocol allowing adaptive corruptions. More details follow in the next sessions.

3.3 Main Building Block: Constructing a Blind Fragmented IBKEM from an IBKEM

Definition and Security Properties of a Blind IBKEM Scheme. Following [BKP14], we recalled in Section 2.2 page 4 the definitions, notations and security properties for an IBE scheme, seen as an Identity-Based Key Encapsulation (IBKEM) scheme. We continue to follow the KEM formalism by adapting the definition of a Blind IBE scheme given in [GH07] to this setting.

Definition 7 (Blind Identity-Based Key Encapsulation Scheme).
A Blind Identity-Based Key Encapsulation scheme $\mathsf{BlindIBKEM}$ *consists of four PPT algorithms* $(\mathsf{Gen}, \mathsf{BlindUSKGen}, \mathsf{Enc}, \mathsf{Dec})$ *with the following properties:*

- Gen, Enc *and* Dec *are defined as for a traditional* IBKEM *scheme.*
- $\mathsf{BlindUSKGen}(\langle(\mathcal{S},\mathsf{msk})(\mathcal{U},\mathsf{id},\ell;\boldsymbol{\rho})\rangle)$ *is an interactive protocol, in which an honest user* $\mathcal{U}$ *with identity* $\mathsf{id} \in \mathcal{ID}$ *obtains the corresponding user secret key* $\mathsf{usk}[\mathsf{id}]$ *from the master authority* $\mathcal{S}$ *or outputs an error message, while* $\mathcal{S}$*'s output is nothing or an error message (*ℓ *is a label and* $\boldsymbol{\rho}$ *the randomness).*

Defining the security of a BlindIBKEM requires two additional properties, stated as follows (see [GH07, pages 6 and 7] for the formal security games):

1. **Leak-free Secret Key Generation** (called Leak-free Extract for Blind IBE security in the original paper): A potentially malicious user cannot learn anything by executing the BlindUSKGen protocol with an honest authority which he could not have learned by executing the USKGen protocol with an honest authority; Moreover, as in USKGen, the user must know the identity for which he is extracting a key.

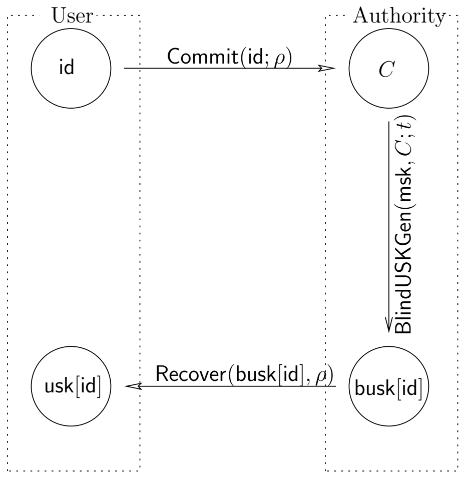
2. **Selective-failure Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol; Moreover, the authority cannot cause the BlindUSKGen protocol to fail in a manner dependent on the user's choice.
   For our applications, we only need a weakened property for blindness:[8]
3. **Weak Blindness**: A potentially malicious authority cannot learn anything about the user's choice of identity during the BlindUSKGen protocol.

High-Level Idea of the Transformation. We now show how to obtain a BlindIBKEM scheme from any IBKEM scheme. From a high-level point of view, this transformation mixes two pre-existing approaches.

First, we are going to consider a reverse Naor transform [BF01, CFH+07]: He drew a parallel between Identity-Based Encryption schemes and signature schemes, by showing that a user secret key on an identity can be viewed as the signature on this identity, the verification process therefore being a test that any chosen valid ciphertext for the said identity can indeed be decrypted using the *signature* scheme.

Then, we are going to use Fischlin [Fis06] round-optimal approach to blind signatures, where the whole interaction is done in one pass: First, the user commits to the message, then he recovers a signature linked to his commitment. For sake of simplicity, instead of using a Non-Interactive Zero-Knowledge Proof of Knowledge of a signature, we are going to follow the [BFPV10, BPV12] approach, where thanks to an additional term, the user can extract a signature on the identity from a signature on the committed identity.

Omitting technical details described more precisely in the following sections, the main idea of the transformation of the IBKEM scheme in order to blind a user key request is described in Figure 3.



1. A user commits to the targeted identity id using some randomness ρ.
2. The authority possesses an algorithm allowing it to generate keys for committed identities using its master secret key msk, and some randomness t, in order to obtain a blinded user secret key busk[id].
3. The user using solely the randomness used in the initial commitment is able to recover the requested secret key from the authority's generated value.

Fig. 3. Generic Transformation of an IBKEM into a Blind IBKEM (naive approach)

Generic Transformation of an IBKEM into a Blind IBKEM. It now remains to explain how one can fulfill the idea highlighted in Figure 3. The technique to blind a user key request uses a smooth projective hash function (see Section 2.3), and is often called *implicit decommitment* in recent works: the IBKEM secret key is sent hidden in such a way that it can only be recovered if the

[8] Two things to note: First, Selective Failure would be considered as a Denial of Service in the Oblivious Transfer setting. Then, we do not restrict ourselves to schemes where the blindness adversary has access to the generated user keys, as reliable erasures in the OT protocol provide us a way to forget them before being corrupted (otherwise we would need to use a randomizable base IBE).

user knows how to open the initial commitment on the correct identity. We assume the existence of a labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cca}}, \mathsf{KeyGen}_{\mathrm{cca}}, \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}, \mathsf{Decrypt}^{\ell}_{\mathrm{cca}})$ compatible with an SPHF defined by $(\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ onto a set G (where ℓ is a label defined by the global protocol). By "compatible", we mean that the SPHF can be defined over a language $\mathfrak{L}^c_{\mathsf{id}} \subset X$, where $\mathfrak{L}^c_{\mathsf{id}} = \{C \mid \exists \boldsymbol{\rho} \text{ such that } C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$. In the KeyGen algorithm, the description of the language $\mathfrak{L}_{\mathsf{id}} = \{\mathsf{id}\}$ thus implicitly defines the language $\mathfrak{L}^c_{\mathsf{id}}$ of CCA-encryptions of elements of $\mathfrak{L}_{\mathsf{id}}$. We additionally use a key derivation function $\mathtt{KDF}$ to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in G$. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma just lead to a security loss linear in the number of extractions. This gives the following protocol for $\mathsf{BlindUSKGen}$, described in Figure 4.

- The user computes an encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(\mathsf{id}; \boldsymbol{\rho})\}$.
- For every identity id', the server computes the key $\mathsf{usk}[\mathsf{id}']$ along with a pair of (secret, public) hash keys $(\mathsf{hk}_{\mathsf{id}'}, \mathsf{hp}_{\mathsf{id}'})$ for a smooth projective hash function on the language $\mathfrak{L}^c_{\mathsf{id}'}$:
  $\mathsf{hk}_{\mathsf{id}'} = \mathsf{HashKG}(\ell, \mathfrak{L}^c_{\mathsf{id}'}, \mathsf{param})$ and $\mathsf{hp}_{\mathsf{id}'} = \mathsf{ProjKG}(\mathsf{hk}_{\mathsf{id}'}, \ell, (\mathfrak{L}^c_{\mathsf{id}'}, \mathsf{param}), \mathsf{id}')$.
  He also compute the corresponding hash value
  $H_{\mathsf{id}'} = \mathsf{Hash}(\mathsf{hk}_{\mathsf{id}'}, (\mathfrak{L}^c_{\mathsf{id}'}, \mathsf{param}), (\ell, C))$.
  Finally, he sends $(\mathsf{hp}_{\mathsf{id}'}, \mathsf{usk}[\mathsf{id}'] \oplus \mathtt{KDF}(H_{\mathsf{id}'}))$ for every id', where $\oplus$ is a compatible operation.
- Thanks to $\mathsf{hp}_{\mathsf{id}}$, the user is able to compute the corresponding projected hash value $H'_{\mathsf{id}} = \mathsf{ProjHash}(\mathsf{hp}_{\mathsf{id}}, (\mathfrak{L}^c_{\mathsf{id}}, \mathsf{param}), (\ell, C), \boldsymbol{\rho})$. He then recovers $\mathsf{usk}[\mathsf{id}]$ for the initially committed identity id since $H_{\mathsf{id}} = H'_{\mathsf{id}}$.

Fig. 4. Summary of the Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$ for a blind IBE

Theorem 8. *If* IBKEM *is a* $\mathsf{PR\text{-}ID\text{-}CPA}$*-secure identity-based key encapsulation scheme and* $\mathcal{E}$ *a labeled* CCA*-encryption scheme compatible with an* SPHF*, then* $\mathsf{BlindIBKEM}$ *is leak free and weak blind.*

Proof. First, $\mathsf{BlindIBKEM}$ satisfies leak-free secret key generation since it relies on the CCA security on the encryption scheme, forbidding a user to open it to another identity than the one initially encrypted. Furthermore, the pseudo-randomness of the SPHF ensures that the blinded user key received for id is indistinguishable from random if he encrypted $\mathsf{id}' \neq \mathsf{id}$. Finally, the weak blindness also relies on the CCA security on the encryption scheme, since an encryption of id is indistinguishable from a encryption of $\mathsf{id}' \neq \mathsf{id}$. $\square$

Using a Blind IBKEM in our Application to Adaptive Oblivious Transfer. The previous approach allows to transform an IBKEM into a Blind IBKEM, but it has a huge drawback in our context: Since we assume an exponential identity space, it requires an exponential number of answers from the authority, which cannot help us to fulfill logarithmic complexity in our application. However, if we focus on the special case of affine IBE with bitwise function[9], a user key can be described as the list $(\mathsf{usk}[0], \mathsf{usk}[0, \mathsf{id}_0], \ldots, \mathsf{usk}[m-1, \mathsf{id}_{m-1}])$ if id_i is the i-th bit of the identity id. One can thus manage to be much more efficient by sending each "bit" evaluation on the user secret key, hidden with a smooth projective hash value on the language "the i-th bit of the identity is a 0 (or 1)", which is common to all identities. We can thus reduce the number of languages from the number of identities (which is exponential) to the length of an identity (which is polynomial). For security reasons, one cannot give directly the evaluation value, but as

[9] They were defined in [BKP14]. Affine IBE derive their name from the fact that only affine operations are done on the identity bits (no hashing, square rooting, inverting... are allowed).

we are considering the sum of the evaluations for each bit, we simply add a Shamir-like secret sharing, by adding randomness that is going to cancel out at the end.

- The user computes a bit-per-bit encryption of the expected identity id and keeps the randomness ρ: $C = \mathsf{Encrypt}^{\ell}_{\mathsf{cca}}(\mathsf{id};\boldsymbol{\rho})\}$.
- The server computes a fragmented version of all the keys $\mathsf{usk}[\mathsf{id}']$, *i.e.* all the values $\mathsf{usk}[i,b]$ for i from 0 up to the length m of the keys and $b \in \{0,1\}$. He also computes a pair of (secret, public) hash keys $(\mathsf{hk}_{i,b}, \mathsf{hp}_{i,b})$ for a smooth projective hash function on the language $\mathfrak{L}^c_{i,b}$: "The i-th bit of the value encrypted into C is b", *i.e.* $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\ell, \mathfrak{L}^c_{i,b}, \mathsf{param})$ and $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \ell, (\mathfrak{L}^c_{i,b}, \mathsf{param}))$. He also computes the corresponding hash value $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, (\mathfrak{L}^c_{i,b}, \mathsf{param}), (\ell, C))$ and chooses random values z_i. Finally, he sends, for each (i,b), $(\mathsf{hp}_{i,b}, \mathsf{busk}[i,b])$, where $\mathsf{busk}[i,b] = \mathsf{usk}[i,b] \oplus \mathtt{KDF}(H_{i,b}) \oplus z_i$, together with $Z = \mathsf{usk}_0 \ominus \left(\bigoplus_i z_i\right)$, where $\oplus$ is a compatible operation and $\ominus$ its inverse.
- Thanks to the $\mathsf{hp}_{i,\mathsf{id}_i}$ for the initially committed identity id, the user is able to compute the corresponding projected hash value
$$H'_{i,\mathsf{id}_i} = \mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, (\mathfrak{L}^c_{i,\mathsf{id}_i}, \mathsf{param}), (\ell, C), \boldsymbol{\rho}),$$
that should be equal to H_{i,id_i} for all i. From the values $\mathsf{busk}[i,\mathsf{id}_i]$, he then recovers $\mathsf{usk}[i,\mathsf{id}_i] \oplus z_i$. Finally, with the operation $\left(\bigoplus_i (\mathsf{usk}[i,\mathsf{id}_i] \oplus z_i)\right) \oplus Z$, he recovers the expected $\mathsf{usk}[\mathsf{id}]$.

Fig. 5. Summary of the Generic Construction of $\mathsf{BlindUSKGen}(\langle(\mathcal{S},\mathsf{msk})(\mathcal{U},\mathsf{id},\ell;\boldsymbol{\rho})\rangle)$ for a Blind affine IBE

As a last step, we finally need to make our construction compatible with the UC framework with adaptive corruptions. In this context, interactions should make sense for any possible input chosen by the environment and learnt a posteriori in the simulation during the corruption of an honest party. From the user side, this implies that the last flow should contain enough recoverable information so that a simulator, having sent a commitment to an incorrect identity, can extract the proper user secret key corresponding to the correct identity recovered after the corruption. From the server side, this implies that the IBKEM scheme is defined such as one is able to adapt the user secret keys in order to correspond to the new database learnt a posteriori. Of course, not all schemes allow this property, but this will be the case in the pairing scenario considered in our concrete instantiation.

To deal with corruptions of the user, recall that a simulated server (knowing the secret key of the encryption scheme) is already able to extract the identity committed to. But we now consider that, for all id, $\mathfrak{L}_{\mathsf{id}}$ is the language of the equivocable commitments on words in the inner language $\widetilde{\mathfrak{L}}_{\mathsf{id}} = \{\mathsf{id}\}$. We assume them to be a *Trapdoor Collection of Languages*, which means that it is computationally hard to sample an element in $\mathfrak{L}_1 \cap \cdots \cap \mathfrak{L}_n$, except for the simulator, who possesses a trapdoor tk (the equivocation trapdoor) allowing it to sample an element in the intersection of languages. This allows a simulated user (knowing this trapdoor) not to really bind to any identity during the commitment phase. The only difference with the algorithm described in Figure 5 is that the user now encrypts this word W (which is an equivocable commitment on his identity id) rather than directly encrypting his identity id: $C = \mathsf{Encrypt}^{\ell}_{\mathsf{cca}}(W;\boldsymbol{\rho})$. This technique is also explained as an application of our OLBE framework, in Appendix F.2. We will directly prove this protocol during the proof of the oblivious transfer scheme.

3.4 Generic Construction of Adaptive OT

We derive from here our generic construction of OT (depicted in Figure 6). We additionally assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an $\mathsf{IND\text{-}CPA}$ encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. First, the owner of the database generates the keys for such an IBE scheme, and encrypts each line i of the database for the identity i. Then when a user wants to request a given line, he runs the blind user key generation algorithm and recovers the key for the expected given line. This leads to the following security result, proven in Appendix D.

Theorem 9. *Assuming that* $\mathsf{BlindUSKGen}$ *is constructed as described above, the adaptive Oblivious Transfer protocol described in Figure 6 UC-realizes the functionality* $\mathcal{F}_{OT}^{\mathfrak{L}}$ *presented in Figure 2 with adaptive corruptions assuming reliable erasures.*

CRS generation:
$\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.
Database Preparation:

1. Server runs $\mathsf{Gen}(\mathfrak{K})$, to obtain $\mathsf{mpk}, \mathsf{msk}$.
2. For each line t, he computes $(D_t, K_t) = \mathsf{Enc}(\mathsf{mpk}, t)$, and $L_t = K_t \oplus DB(t)$.
3. He also computes $\mathsf{usk}[i, b]$ for all $i = 1 \ldots, m$ and $b = 0, 1$ and erases msk.
4. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
5. He then publishes $\mathsf{mpk}, \{(D_t, L_t)\}_t, \mathsf{pk}$.

Index query on s:

1. User chooses a random value S, computes $R \leftarrow F(S)$ and encrypts S under pk:
   $c \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, S)$
2. User computes $\mathcal{C}$ with the first flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{U}, \mathcal{S})$ (see Figure 5).
3. User stores the random $\boldsymbol{\rho}_s = \{\boldsymbol{\rho}_*\}$ needed to open $\mathcal{C}$ to s, and completely erases the rest, including the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $(c, \mathcal{C})$ to the Server

IBE input msk:

1. Server decrypts $S \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $R \leftarrow F(S)$
2. Server runs the second flow of $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, s, \ell; \boldsymbol{\rho})\rangle)$ on $\mathcal{C}$ (see Figure 5).
3. Server erases every new value except $(\mathsf{hp}_{i,b})_{i,b}, (\mathsf{busk}[i, b])_{i,b}, Z \oplus R$ and sends them over a secure channel.

Data recovery:

1. User then using, $\boldsymbol{\rho}_s$ recovers $\mathsf{usk}[s]$ from the values received from the server.
2. He can then recover the expected information with $\mathsf{Dec}(\mathsf{usk}[s], s, D_s) \oplus L_s$ and erases everything else.

Fig. 6. Adaptive UC-Secure 1-out-of-n OT from a Fragmented Blind IBE

3.5 Pairing-Based Instantiation of Adaptive OT

Affine Bit-Wise Blind IBE. In [BKP14], the authors propose a generic framework to move from affine Message Authentication Code to IBE, and they propose a tight instantiation of such a MAC, giving an affine bit-wise IBE, which seems like a good candidate for our setting (making it blind and fragmented).

We are thus going to use the family of IBE described in the following picture (Figure 7), which is their instantiation derived from a Naor-Reingold MAC[10]. In the following, $h_i()$ are injective deterministic public functions mapping a bit to a scalar in $\mathbb{Z}_p$.

A property that was not studied in this paper was the blind user key generation: How to generate and answer blind user secret key queries? We answer to this question by proposing the $k - \mathsf{MDDH}$-based variation presented in Figure 8. To fit the global framework we are going to consider the equivocable language of each chameleon hash of the identity bits $(\boldsymbol{a}_i, \boldsymbol{b}_{i,m_i})$, and then a Cramer-Shoup like encryption of $\boldsymbol{b}$ into $\boldsymbol{d}$ (more details in Section B.2). We denote this process as Har in the following protocol, and by $\mathfrak{L}_{\mathsf{Har},i,\mathsf{id}_i}$ the language on identity bits. We thus obtain the following security results.

Theorem 10. *This construction achieves both the weak Blindness, and the leak-free secret key generation requirements under the* $k - \mathsf{MDDH}$ *assumption.*

[10] For the reader familiar with the original result, we combine $x, \boldsymbol{y}$ into a bigger $\boldsymbol{y}$ to lighten the notations, and compact the (x'_i, y'_i) values into a single y' as this has no impact on their construction.

```
$\mathsf{Gen}(\mathfrak{K})$:
$\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{B} = \overline{\boldsymbol{A}}$
For $i \in [\![0, \ell]\!] : \boldsymbol{Y}_i \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \boldsymbol{Z}_i = \boldsymbol{Y}_i^\top \cdot \boldsymbol{A} \in \mathbb{Z}_p^k$
$\boldsymbol{y}' \xleftarrow{\$} \mathbb{Z}_p^{k+1}; \boldsymbol{z}' = \boldsymbol{y}'^\top \cdot \boldsymbol{A} \in \mathbb{Z}_q^k$
$\mathsf{mpk} := (\mathcal{G}, [\boldsymbol{A}]_1, ([\boldsymbol{Z}_i]_1)_{i \in [\![0,\ell]\!]}, [\boldsymbol{z}']_1)$
$\mathsf{msk} := (\boldsymbol{Y}_i)_{i \in [\![0,\ell]\!]}, \boldsymbol{y}'$
Return $(\mathsf{mpk}, \mathsf{msk})$

$\mathsf{USKGen}(\mathsf{msk}, \mathsf{id})$:
$\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{t} = \boldsymbol{B}\boldsymbol{s}$
$\boldsymbol{w} = (\boldsymbol{Y}_0 + \sum_{i=1}^{\ell} h_i(\mathsf{id}_i)\boldsymbol{Y}_i)\boldsymbol{t} + \boldsymbol{y}' \in \mathbb{Z}_p^{k+1}$
Return $\mathsf{usk}[\mathsf{id}] := ([\boldsymbol{t}]_2, [\boldsymbol{w}]_2) \in \mathbb{G}_2^{k+k+1}$
```

```
$\mathsf{Enc}(\mathsf{mpk}, \mathsf{id})$:
$\boldsymbol{r} \xleftarrow{\$} \mathbb{Z}_p^k$
$\boldsymbol{c}_0 = \boldsymbol{A}\boldsymbol{r} \in \mathbb{Z}_p^{k+1}$
$\boldsymbol{c}_1 = (\boldsymbol{Z}_0 + \sum_{i=1}^{\ell} h_i(\mathsf{id}_i)\boldsymbol{Z}_i) \cdot \boldsymbol{r} \in \mathbb{Z}_p$
$K = \boldsymbol{z}' \cdot \boldsymbol{r} \in \mathbb{Z}_p$.
Return $[K]_T$
and $\mathsf{C} = ([\boldsymbol{c}_0]_1, [\boldsymbol{c}_1]_1) \in \mathbb{G}_1^{k+1+1}$

$\mathsf{Dec}(\mathsf{usk}[\mathsf{id}], \mathsf{id}, \mathsf{C})$:
Parse $\mathsf{usk}[\mathsf{id}] = ([\boldsymbol{t}]_2, [\boldsymbol{w}]_2)$
Parse $\mathsf{C} = ([\boldsymbol{c}_0]_1, [\boldsymbol{c}_1]_1)$
$K = e([\boldsymbol{c}_0]_1, [\boldsymbol{w}]_2) \cdot e([\boldsymbol{c}_1]_1, [\boldsymbol{t}]_2)^{-1}$
Return $K \in \mathbb{G}_T$
```

Fig. 7. A fragmentable affine IBKEM.

```
– First flow: $\mathcal{U}$ starts by computing
    $\boldsymbol{\rho} \xleftarrow{\$} \mathbb{Z}_p^{1+4\times\ell}$,
    $\boldsymbol{a}, \boldsymbol{d} = \mathsf{Har}(\mathsf{id}, \ell; \rho) \in \mathbb{Z}_p^{\ell} \times \mathbb{Z}_p^{2\times(k+3)\ell}$,
    Sends $\mathcal{C} = ([\boldsymbol{a}]_1, [\boldsymbol{d}]_2)$ to $S$
– Second Flow: $\mathcal{S}$ then proceeds
    $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{t} = \boldsymbol{B}\boldsymbol{s}$, $\boldsymbol{f} \xleftarrow{\$} \mathbb{Z}_p^{\ell\times k+1}$,
    For each $i \in [\![1, \lceil\log n\rceil]\!], b \in [\![0, 1]\!]$:
        $\mathsf{hk}_{i,b} = \mathsf{HashKG}(\mathfrak{L}_{\mathsf{Har},i,b}, \boldsymbol{C})$
        $\mathsf{hp}_{i,b} = \mathsf{ProjKG}(\mathsf{hk}_{i,b}, \mathfrak{L}_{\mathsf{Har},i,b}, \boldsymbol{C})$
        $H_{i,b} = \mathsf{Hash}(\mathsf{hk}_{i,b}, \mathfrak{L}_{\mathsf{Har},i,b}, \boldsymbol{C})$
        $\boldsymbol{\omega}_{i,b} = (b\boldsymbol{Y}_i)\boldsymbol{t} + \boldsymbol{f}_i + H_{i,b}$
```

```
Then sets $\boldsymbol{w}_0 = \boldsymbol{Y}_0\boldsymbol{t} + \boldsymbol{y}' - \sum_{i=1}^{\ell} \boldsymbol{f}_i \in \mathbb{Z}_p^{k+1}$
Returns $\mathsf{busk} :=$
$([\boldsymbol{t}]_2, [\boldsymbol{w}_0]_2, \{[\boldsymbol{\omega}_{i,b}]_2\}, \{[\mathsf{hp}_{i,b}]_2\})$
  – $\mathsf{BlindUSKGen}_3$: $\mathcal{U}$ then recovers his key
    For each $i \in [\![1, \ell]\!]$:
        $H'_i =$
    $\mathsf{ProjHash}(\mathsf{hp}_{i,\mathsf{id}_i}, \mathfrak{L}_{\mathsf{Har},i,\mathsf{id}_i}, \boldsymbol{C}, \rho_i)$
        $\boldsymbol{w}_i = \omega_{i,\mathsf{id}_i} - H'_i$
    $\boldsymbol{w} = \boldsymbol{w}_0 + \sum_{i=1}^{\ell} \boldsymbol{w}_i$
    And then recovers $\mathsf{usk}[\mathsf{id}] :=$
    $[\boldsymbol{t}]_2, [\boldsymbol{w}]_2$
```

Fig. 8. $\mathsf{BlindUSKGen}(\langle(\mathcal{S}, \mathsf{msk})(\mathcal{U}, \mathsf{id}, \ell; \boldsymbol{\rho})\rangle)$.

The first one is true under the indistinguishability of the generalized Cramer-Shoup encryption recalled in Section B.2, as the server learns nothing about the line requested during the first flow. It should even be noted that because of the inner chameleon hash, a simulator is able to use the trapdoor to do a commitment to every possible words of the set of languages at once, and so can adaptively decide which id he requested. The proof of the second result is delayed to Appendix C.

For sake of generality, any bit-wise affine IBE could work (like for example Waters IBE [Wat05]), the additional price paid for tightness here is very small and allows to have a better reduction in the proof, but it it not required by the framework itself.

Adaptive UC-Secure Oblivious Transfer. We finally get our instantiation by combining this $k-\mathsf{MDDH}$-based blind IBE with a $k-\mathsf{MDDH}$ variant of El Gamal for the CPA encryption needed. (see Appendix B.2 for details). The requirement on the IBE blind user secret key generation (being able to adapt the key if the line changes) is achieved assuming that the server knows the discrete logarithms of the database lines. This is quite easy to achieve by assuming that for all line s, $DB(s) = [db(s)]_1$ where $db(s)$ is the real line (thus known). It implies a few more computation on the user's side in order to recover $db(s)$ from $DB(s)$, but this remains completely feasible if the lines belong to a small space. For practical applications, one could imagine to split all 256-bit lines into 8 pieces for a decent/constant trade-off in favor of computational efficiency.

For $k = 1$, so under the classical SXDH assumption, the first flow requires $8 \log |DB|$ elements in $\mathbb{G}_1$ for the CCA encryption part and $\log(|DB| + 1)$ in $\mathbb{G}_2$ for the chameleon one, while the second flow would now require $1 + 4 \log |DB|$ elements in $\mathbb{G}_1$, $1 + 2 \log |DB|$ for the fragmented masked key, and $2 \log |DB|$ for the projection keys.

4 Oblivious Language-Based Envelope

The previous construction opens new efficient applications to the already known Oblivious-Transfer protocols. But what happens when someone wants some additional access control by requesting extra properties, like if the user is only allowed to ask two lines with the same parity bits, the user can only request lines for whose number has been signed by an authority, or even finer control provided through credentials?

In this section we propose to develop a new primitive, that we call Oblivious Language-Based Envelope (OLBE). The idea generalizes that of Oblivious Transfer and OSBE, recalled right afterwards, for n messages (with n polynomial in the security parameter $\mathfrak{K}$) to provide the best of both worlds.

4.1 Oblivious Signature-Based Envelope

We recall the definition and security requirements of an OSBE protocol given in [LDB03,BPV12], in which a sender $\mathcal{S}$ wants to send a private message $m \in \{0,1\}^{\mathfrak{K}}$ to a recipient $\mathcal{R}$ in possession of a valid certificate/signature on a public message M (given by a certification authority).

Definition 11 (Oblivious Signature-Based Envelope). *An* OSBE *scheme is defined by four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$, *and one interactive protocol* $\mathsf{Protocol}\langle\mathcal{S}, \mathcal{R}\rangle$*:*
- $\mathsf{Setup}(1^{\mathfrak{K}})$, *where $\mathfrak{K}$ is the security parameter, generates the global parameters* param*;*
- $\mathsf{KeyGen}(\mathfrak{K})$ *generates the keys* $(\mathsf{vk}, \mathsf{sk})$ *of the certification authority;*
- $\mathsf{Sign}(\mathsf{sk}, M)$ *produces a signature σ on the input message M, under the signing key* sk*;*
- $\mathsf{Verify}(\mathsf{vk}, M, \sigma)$ *checks whether σ is a valid signature on M, w.r.t. the public key* vk*; it outputs 1 if the signature is valid, and 0 otherwise.*
- $\mathsf{Protocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ *between the sender $\mathcal{S}$ with the private message P, and the recipient $\mathcal{R}$ with a certificate σ. If σ is a valid signature under* vk *on the common message M, then $\mathcal{R}$ receives m, otherwise it receives nothing. In any case, $\mathcal{S}$ does not learn anything.*

The authors of [BPV12] proposed some variations to the original definitions from [LDB03], in order to prevent some interference by the authority. Following them, an OSBE scheme should fulfill the following security properties. The formal security games are given in [BPV12]. No UC functionality has already been given, to the best of our knowledge.

– *correct*: the protocol actually allows $\mathcal{R}$ to learn P, whenever σ is a valid signature on M under vk;
– *semantically secure*: the recipient learns nothing about $\mathcal{S}$'s input m if it does not use a valid signature σ on M under vk as input. More precisely, if $\mathcal{S}_0$ owns P_0 and $\mathcal{S}_1$ owns P_1, the recipient that does not use a valid signature cannot distinguish an interaction with $\mathcal{S}_0$ from an interaction with $\mathcal{S}_1$ even if he has eavesdropped on several interactions $\langle \mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ with valid signatures, and the same sender's input P;
– *escrow-free* (*oblivious with respect to the authority*): the authority (owner of the signing key sk), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a valid signature σ on M under vk as input.
– *semantically secure w.r.t. the authority*: after the interaction, the authority (owner of the signing key sk) learns nothing about m from a passive access to a challenge transcript.

4.2 Definition of an Oblivious Language-Based Envelope

In such a protocol, a sender $\mathcal{S}$ wants to send one or several private messages (up to $n_{\max} \leq n$) among $(m_1, \ldots, m_n) \in (\{0,1\}^\ell)^n$ to a recipient $\mathcal{R}$ in possession of a tuple of words $W = (W_{i_1}, \ldots, W_{i_{n_{\max}}})$ such that some of the words W_{i_j} may belong to the corresponding language $\mathfrak{L}_{i_j}$. More precisely, the receiver gets each m_{i_j} as soon as $W_{i_j} \in \mathfrak{L}_{i_j}$ with the requirement that he gets at most $n_{\max}$ messages. In such a scheme, the languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ are assumed to be a trapdoor collection of languages, publicly verifiable and self-randomizable (see Section 2.3 for the definitions of the properties of the languages).

The collections of words can be a single certificate/signature on a message M (encompassing OSBE, with $n = n_{\max} = 1$), a password, a credential, a line number (encompassing 1-out-of-n oblivious transfer[11], with $n_{\max} = 1$), k line numbers (encompassing k-out-of-n oblivious transfer, with $n_{\max} = k$), etc. (see Section F for detailed examples). Following the definitions for OSBE recalled above and given in [LDB03, BPV12], we give the following definition for OLBE. As we consider simulation-based security (in the UC framework), we allow a simulated setup SetupT to be run instead of the classical setup Setup in order to allow the simulator to possess some trapdoors. Those two setup algorithms should be indistinguishable.

Definition 12 (Oblivious Language-Based Envelope). *An* OLBE *scheme is defined by four algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Samp}, \mathsf{Verify})$*, and one interactive protocol* $\mathsf{Protocol}\langle \mathcal{S}, \mathcal{R}\rangle$*:*

– $\mathsf{Setup}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param*, among which the numbers* n *and* $n_{\max}$*;*
or $\mathsf{SetupT}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, additionally allows the existence*[12] *of a trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*
– $\mathsf{KeyGen}(\mathsf{param}, \mathfrak{K})$ *generates, for all* $i \in \{1, \ldots, n\}$*, the description of the language* $\mathfrak{L}_i$ *(as well as the language key* $\mathsf{sk}_{\mathfrak{L}_i}$ *if need be). If the parameters* param *were defined by* SetupT*, this implicitly also defines the common trapdoor* tk *for the collection of languages* $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$*.*
– $\mathsf{Samp}(\mathsf{param}, I)$ *or* $\mathsf{Samp}(\mathsf{param}, I, (\mathsf{sk}_{\mathfrak{L}_i})_{i \in I})$ *such that* $I \subset \{1, \ldots, n\}$ *and* $|I| = n_{\max}$*, generates a list of words* $(W_i)_{i \in I}$ *such that* $W_i \in \mathfrak{L}_i$ *for all* $i \in I$*;*
– $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ *checks whether* W_i *is a valid word in the language* $\mathfrak{L}_i$*. It outputs* 1 *if the word is valid,* 0 *otherwise;*

[11] Even if, as explained in the former section, we would rather consider equivocable commitments of line numbers than directly line numbers, in order to get adaptive UC security.
[12] The specific trapdoor will depend on the languages and be computed in the KeyGen algorithm.

– $\mathsf{Protocol}\langle\mathcal{S}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(m_1,\ldots,m_n)),\mathcal{R}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(W_i)_{i\in I})\rangle$, *which is executed between the sender* $\mathcal{S}$ *with the private messages* $(m_1,\ldots,P_n)$ *and corresponding languages* $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$, *and the recipient* $\mathcal{R}$ *with the same languages and the words* $(W_i)_{i\in I}$ *with* $I\subset\{1,\ldots,n\}$ *and* $|I|=n_{\max}$*, proceeds as follows. For all* $i\in I$*, if the algorithm* $\mathsf{Verify}_i(W_i,\mathfrak{L}_i)$ *returns* 1*, then* $\mathcal{R}$ *receives* m_i*, otherwise it does not. In any case,* $\mathcal{S}$ *does not learn anything.*

4.3 Security Properties and Ideal Functionality of OLBE

Since we aim at proving the security in the universal composability framework, we now describe the corresponding ideal functionality (depicted in Figure 9). However, in order to ease the comparison with an OSBE scheme, we first list the security properties required, following [LDB03] and [BPV12]:

– *correct*: the protocol actually allows $\mathcal{R}$ to learn $(m_i)_{i\in I}$, whenever $(W_i)_{i\in I}$ are valid words of the languages $(\mathfrak{L}_i)_{i\in I}$, where $I\subset\{1,\ldots,n\}$ and $|I|=n_{\max}$;
– *semantically secure* (*sem*): the recipient learns nothing about the input m_i of $\mathcal{S}$ if it does not use a word in $\mathfrak{L}_i$. More precisely, if $\mathcal{S}_0$ owns $m_{i,0}$ and $\mathcal{S}_1$ owns $m_{i,1}$, the recipient that does not use a word in $\mathfrak{L}_i$ cannot distinguish between an interaction with $\mathcal{S}_0$ and an interaction with $\mathcal{S}_1$ even if the receiver has seen several interactions
$$\langle\mathcal{S}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(m_1,\ldots,m_n)),\mathcal{R}((\mathfrak{L}_1,\ldots,\mathfrak{L}_n),(W'_j)_{j\in I})\rangle$$
with valid words $W'_i\in\mathfrak{L}_i$, and the same sender's input m_i;
– *escrow free* (*oblivious with respect to the authority*): the authority corresponding to the language $\mathfrak{L}_i$ (owner of the language secret key $\mathsf{sk}_{\mathfrak{L}_i}$ – if it exists), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a word W_i in the language $\mathfrak{L}_i$ or not. This requirement also holds for anyone holding the trapdoor key tk.
– *semantically secure w.r.t. the authority* (*sem**): after the interaction, the trusted authority (owner of the language secret keys if they exist) learns nothing about the values $(m_i)_{i\in I}$ from the transcript of the execution. This requirement also holds for anyone holding the trapdoor key tk.

Moreover, the Setups should be indistinguishable and it should be infeasible to find a word belonging to two or more languages without the knowledge of tk.

The functionality $\mathcal{F}_{\mathrm{OLBE}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1,\ldots,\mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1,\ldots,\mathfrak{P}_N$ via the following queries:

– **Upon receiving from party $\mathfrak{P}_i$ an input of the form $(\mathtt{Send},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(m_1,\ldots,m_n))$** , with $m_k\in\{0,1\}^{\mathfrak{K}}$ for all k: record the tuple $(\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(m_1,\ldots,m_n))$ and reveal $(\mathtt{Send},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further $\mathtt{Send}$-message with the same ssid from $\mathfrak{P}_i$.
– **Upon receiving an input of the form $(\mathtt{Receive},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(W_i)_{i\in I})$ where $I\subset\{1,\ldots,n\}$ and $|I|=n_{\max}$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(m_1,\ldots,m_n))$ is not recorded. Otherwise, reveal $(\mathtt{Receive},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send the message $(\mathtt{Received},\mathsf{sid},\mathsf{ssid},\mathfrak{P}_i,\mathfrak{P}_j,(m'_k)_{k\in I})$ to $\mathfrak{P}_j$ where $m'_k=m_k$ if $\mathsf{Verify}_k(W_k,\mathfrak{L}_k)$ returns 1, and $m'_k=\perp$ otherwise. Ignore further $\mathtt{Received}$-message with the same ssid from $\mathfrak{P}_j$.

Fig. 9. Ideal Functionality for Oblivious Language-Based Envelope $\mathcal{F}_{\mathrm{OLBE}}$

The ideal functionality is parametrized by a set of languages $(\mathfrak{L}_1,\ldots,\mathfrak{L}_n)$. Since we show in the following sections that one can see OSBE and OT as special cases of OLBE, it is inspired from the oblivious transfer functionality given in [Can01,CKWZ13,ABB+13] in order to provide a framework consistent with works well-known in the literature. As for oblivious transfer (Figure 2), we adapt them to the simple UC framework for simplicity (this enables us to get rid of `Sent` and `Received` queries from the adversary since the delayed outputs are automatically considered in this simpler framework: We implicitly let the adversary determine if it wants to acknowledge the fact that a message was indeed sent). The first step for the sender (`Send` query) consists in telling the functionality he is willing to take part in the protocol, giving as input his intended receiver and the messages he is willing to send (up to $n_{\max}$ messages). For the receiver, the first step (`Receive` query) consists in giving the functionality the name of the player he intends to receive the messages from, as well as his words. If the word does belong to the language, the receiver recovers the sent message, otherwise, he only gets a special symbol $\perp$.

4.4 Generic UC-Secure Instantiation of OLBE with Adaptive Security

For the sake of clarity, we now concentrate on the specific case where $n_{\max} = 1$. This is the most classical case in practice, and suffices for both OSBE and 1-out-of-n OT. In order to get a generic protocol in which $n_{\max} > 1$, one simply has to run $n_{\max}$ protocols in parallel. This modifies the algorithms Samp and Verify as follows: $\mathsf{Samp}(\mathsf{param}, \{i\})$ or $\mathsf{Samp}(\mathsf{param}, \{i\}, \{\mathsf{sk}_{\mathfrak{L}_i}\})$ generates a word $W = W_i \in \mathfrak{L}_i$ and $\mathsf{Verify}_j(W, \mathfrak{L}_j)$ checks whether W is a valid word in $\mathfrak{L}_j$.

Let us introduce our protocol OLBE: we will call $\mathcal{R}$ the receiver and $\mathcal{S}$ the sender. If $\mathcal{R}$ is an honest receiver, then he knows a word $W = W_i$ in one of the languages $\mathfrak{L}_i$. If $\mathcal{S}$ is an honest sender, then he wants to send up a message among $(m_1, \dots, m_n) \in (\{0,1\}^{\mathfrak{K}})^n$ to $\mathcal{R}$. We assume the languages $\mathfrak{L}_i$ to be self-randomizable and publicly verifiable. We also assume the collection of languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$ possess a trapdoor, that the simulator is able to find by programming the common reference string. As recalled in the previous section, this trapdoor enables him to find a word lying in the intersection of the n languages. This should be infeasible without the knowledge of the trapdoor. Intuitively, this allows the simulator to commit to all languages at once, postponing the time when it needs to choose the exact language he wants to bind to. On the opposite, if a user was granted the same possibilities, this would prevent the simulator to extract the chosen language.

We assume the existence of a labeled CCA-encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cca}}, \mathsf{KeyGen}_{\mathrm{cca}}, \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}, \mathsf{Decrypt}^{\ell}_{\mathrm{cca}})$ compatible with an SPHF onto a set G. In the KeyGen algorithm, the description of the languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$ thus implicitly defines the languages $(\mathfrak{L}^c_1, \dots, \mathfrak{L}^c_n)$ of CCA-encryptions of elements of the languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$. We additionally use a key derivation function `KDF` to derive a pseudo-random bit-string $K \in \{0,1\}^{\mathfrak{K}}$ from a pseudo-random element $v \in G$. One can use the Leftover-Hash Lemma [HILL99], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash Lemma just lead to a security loss linear in the number of extractions. We also assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and an IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter.

We follow the ideas of the oblivious transfer constructions given in [ABB+13, BC15], giving the protocol presented on Figure 10. For the sake of simplicity, we only give the version for adaptive security, in which the sender generates a public key pk and ciphertext c to create a somewhat secure channel (they would not be used in the static version).

Theorem 13. *The oblivious language-based envelope scheme described in Figure 10 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures, an* IND-CPA *encryption scheme, and an* IND-CCA *encryption scheme admitting an* SPHF *on the language of valid ciphertexts of elements of* $\mathfrak{L}_i$ *for all* i*, as soon as the languages are self-randomizable, publicly-verifiable and admit a common trapdoor. The proof is given in Appendix E.*

4.5 Oblivious Primitives Obtained by the Framework

Classical oblivious primitives such as Oblivious Transfer (both 1-out-of-n and k-out-of-n) or Oblivious Signature-Based Envelope directly lie in this framework and can be seen as examples

CRS: $\mathsf{param} \xleftarrow{\$} \mathsf{Setup}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cca}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cca}}(1^{\mathfrak{K}})$, $\mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.
Pre-flow:

1. Sender generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen.
2. Sender sends pk to User.

Flow From the Receiver $\mathcal{R}$:

1. User chooses a random value J, computes $R \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$.
2. User computes $C \xleftarrow{\$} \mathsf{Encrypt}^{\ell}_{\mathrm{cca}}(W; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$.
3. User completely erases J and the random coins used by $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends C and c to Sender. He also checks the validity of his words: the receiver only keeps the random coins used by $\mathsf{Encrypt}_{\mathrm{cca}}$ for the j such that $\mathsf{Verify}_j(W, \mathfrak{L}_j) = 1$ (since he knows they will be useless otherwise).

Flow From the Sender $\mathcal{S}$:

1. Sender decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and then $R \leftarrow F(J)$.
2. For all $j \in \{1, \ldots, n\}$, sender computes $\mathsf{hk}_j = \mathsf{HashKG}(\ell, \mathfrak{L}^c_j, \mathsf{param})$, $\mathsf{hp}_j = \mathsf{ProjKG}(\mathsf{hk}_j, \ell, (\mathfrak{L}^c_j, \mathsf{param}))$, $v_j = \mathsf{Hash}(\mathsf{hk}_j, (\mathfrak{L}^c_j, \mathsf{param}), (\ell, C))$, $Q_j = m_j \oplus \mathtt{KDF}(v_j) \oplus R$.
3. Sender erases everything except $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$ and sends them over a secure channel.

Message recovery:
Upon receiving $(Q_j, \mathsf{hp}_j)_{j \in \{1,\ldots,n\}}$, $\mathcal{R}$ can recover m_i by computing $m_i = Q_i \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}^c_i, \mathsf{param}), (\ell, C), r) \oplus R$.

Fig. 10. UC-Secure OLBE for One Message (Secure Against Adaptive Corruptions)

of Oblivious Language-Based Envelope. We provide in Appendix F details about how to describe the languages and choose appropriate smooth projective hash functions to readily achieve current instantiations of Oblivious Signature-Based Envelope or Oblivious Transfer from our generic protocol. The framework also enables us to give a new instantiation of Access Controlled Oblivious Transfer under classical assumptions. In such a primitive, the user does not automatically gets the line he asks for, but has to prove that he possesses one of the credential needed to access this particular line.

For the sake of simplicity, all the instantiations given are pairing-based but techniques explained in [BC15] could be used to rely on other families of assumptions, like decisional quadratic residue or even LWE.

Acknowledgments

This work was supported in part by the French ANR EnBid Project (ANR-14-CE28-0003).

References

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, December 2013.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, August 2009.

[AIR01] William Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 119–135. Springer, May 2001.

[AP06] Michel Abdalla and David Pointcheval. A scalable password-based group key exchange protocol in the standard model. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 332–347. Springer, December 2006.

[Att14] Nuttapong Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 557–577. Springer, May 2014.

[BBC$^+$13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, February / March 2013.

[BBC$^+$13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, August 2013.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of uc-secure oblivious transfer. In Tal Malkin, Vladimir Kolesnikov, Allison Bishop Lewko, and Michalis Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 65–86. Springer, 2015.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, August 2001.

[BFPV10] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Rosario Gennaro, editor, *Proceedings of PKC 2011*, Lecture Notes in Computer Science. Springer, 2010. Full version available from the web page of the authors.

[BG13] Stephanie Bayer and Jens Groth. Zero-knowledge argument for polynomial evaluation with application to blacklists. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 646–663. Springer, May 2013.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In Joe Kilian, editor, *TCC 2005*, volume 3378 of *LNCS*, pages 325–341. Springer, February 2005.

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12*, pages 784–796. ACM Press, October 2012.

[BKP14] Olivier Blazy, Eike Kiltz, and Jiaxin Pan. (hierarchical) identity-based encryption from affine message authentication. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 408–425. Springer, August 2014.

[Bon98] Dan Boneh. The decision Diffie-Hellman problem. In *Third Algorithmic Number Theory Symposium (ANTS)*, volume 1423 of *LNCS*. Springer, 1998. Invited paper.

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, March 2012.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CCGS10] Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 255–276. Springer, August 2010.

[CCL15] Ran Canetti, Asaf Cohen, and Yehuda Lindell. A simpler variant of universally composable security for standard multiparty computation. In *CRYPTO 2015, Part II*, LNCS, pages 3–22. Springer, August 2015.

[CDH12] Jan Camenisch, Maria Dubovitskaya, and Kristiyan Haralambiev. Efficient structure-preserving signature scheme from standard assumptions. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12*, volume 7485 of *LNCS*, pages 76–94. Springer, September 2012.

[CDN09] Jan Camenisch, Maria Dubovitskaya, and Gregory Neven. Oblivious transfer with access control. In Ehab Al-Shaer, Somesh Jha, and Angelos D. Keromytis, editors, *ACM CCS 09*, pages 131–140. ACM Press, November 2009.

[CDNZ11] Jan Camenisch, Maria Dubovitskaya, Gregory Neven, and Gregory M. Zaverucha. Oblivious transfer with hidden access control policies. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 192–209. Springer, March 2011.

[CFH$^+$07] Yang Cui, Eiichiro Fujisaki, Goichiro Hanaoka, Hideki Imai, and Rui Zhang. Formal security treatments for signatures from identity-based encryption. In Willy Susilo, Joseph K. Liu, and Yi Mu, editors, *ProvSec 2007*, volume 4784 of *LNCS*, pages 218–227. Springer, November 2007.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *36th FOCS*, pages 41–50. IEEE Computer Society Press, October 1995.

[CKP07] Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 613–630. Springer, August 2007.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, February / March 2013.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CNs07] Jan Camenisch, Gregory Neven, and abhi shelat. Simulatable adaptive oblivious transfer. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 573–590. Springer, May 2007.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, August 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, April / May 2002.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DOR99] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional oblivious transfer and timed-release encryption. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 74–89. Springer, May 1999.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, August 2013.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, August 1984.

[Fis06] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, August 2006.

[GD14] Vandana Guleria and Ratna Dutta. Lightweight universally composable adaptive oblivious transfer. In ManHo Au, Barbara Carminati, and C.-C.Jay Kuo, editors, *Network and System Security*, volume 8792 of *Lecture Notes in Computer Science*, pages 285–298. Springer International Publishing, 2014.

[GH07] Matthew Green and Susan Hohenberger. Blind identity-based encryption and simulatable oblivious transfer. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 265–282. Springer, December 2007.

[GH08] Matthew Green and Susan Hohenberger. Universally composable adaptive oblivious transfer. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *LNCS*, pages 179–197. Springer, December 2008.

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *30th ACM STOC*, pages 151–160. ACM Press, May 1998.

[GKW15] Romain Gay, Iordanis Kerenidis, and Hoeteck Wee. Communication complexity of conditional disclosure of secrets and attribute-based encryption. In *CRYPTO 2015, Part II*, LNCS, pages 485–502. Springer, August 2015.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.

[Har11] Kristiyan Haralambiev. *Efficient Cryptographic Primitives for Non-Interactive Zero-Knowledge Proofs and Applications.* PhD thesis, New York University, 2011.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, August 2007.

[IW14] Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, July 2014.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Efficient oblivious pseudorandom function with applications to adaptive OT and secure computation of set intersection. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *LNCS*, pages 577–594. Springer, March 2009.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, May 2005.

[KLL$^+$15] Aggelos Kiayias, Nikos Leonardos, Helger Lipmaa, Kateryna Pavlyk, and Qiang Tang. Optimal rate private information retrieval from homomorphic encryption. *PoPETs*, 2015(2):222–243, 2015.

[KNP11] Kaoru Kurosawa, Ryo Nojima, and Le Trieu Phong. Generic fully simulatable adaptive oblivious transfer. In Javier Lopez and Gene Tsudik, editors, *ACNS 11*, volume 6715 of *LNCS*, pages 274–291. Springer, June 2011.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, March 2011.

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 182–189. ACM, July 2003.

[LL07] Sven Laur and Helger Lipmaa. A new protocol for conditional disclosure of secrets and its applications. In Jonathan Katz and Moti Yung, editors, *ACNS 07*, volume 4521 of *LNCS*, pages 207–225. Springer, June 2007.

[NP97] Moni Naor and Benny Pinkas. Visual authentication and identification. In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 322–336. Springer, August 1997.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[Ped92] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, August 1992.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, August 2008.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981.

[RKP09] Alfredo Rial, Markulf Kohlweiss, and Bart Preneel. Universally composable adaptive priced oblivious transfer. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 231–247. Springer, August 2009.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, August 1992.

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, August 1984.

[Sha07] Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. `http://eprint.iacr.org/2007/074.pdf`.

[Wat05] Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005.

[Wee14] Hoeteck Wee. Dual system encryption via predicate encodings. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 616–637. Springer, February 2014.

[WHC$^+$14] Xiao Shaun Wang, Yan Huang, T.-H. Hubert Chan, Abhi Shelat, and Elaine Shi. SCORAM: Oblivious RAM for secure computation. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 14*, pages 191–202. ACM Press, November 2014.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th FOCS*, pages 162–167. IEEE Computer Society Press, October 1986.

A Classical Primitives

Digital Signature.

A digital signature scheme $\mathcal{S}$ [DH76, GMR88] allows a signer to produce a verifiable proof that he indeed produced a message. It is de scribed through four algorithms:

Definition 14 (Digital Signature Scheme). $\sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

– $\mathsf{Setup}(1^{\mathfrak{K}})$ *where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param *of the scheme, for example the message space;*

– $\mathsf{KeyGen}(\mathsf{param})$*, outputs a pair of* $(\mathsf{sk}, \mathsf{vk})$*, where* sk *is the (secret) signing key, and* vk *is the (public) verification key;*

– $\mathsf{Sign}(\mathsf{sk}, M; \mu)$*, outputs a signature* $\sigma(M)$*, on a message* M*, under the signing key* sk*, and some randomness* μ*;*

– $\mathsf{Verify}(\mathsf{vk}, M, \sigma)$ *checks the validity of the signature* σ *with respect to the message* M *and the verification key* vk. *And so outputs a bit.*

In the following we will expect at least two properties for signatures:

– *Correctness*: For every pair $(\mathsf{vk}, \mathsf{sk})$ generated by KeyGen, for every message M, and for all randomness μ, we have $\mathsf{Verify}(\mathsf{vk}, M, \mathsf{Sign}(\mathsf{sk}, M; \mu)) = 1$.

– *Existential Unforgeability under Chosen Message Attacks [GMR88]* ($\mathsf{EUF-CMA}$). Even after querying n valid signatures on chosen messages (M_i), an adversary should not be able to output a valid signature on a fresh message M. To formalize this notion, we define a signing oracle OSign:
  - $\mathsf{OSign}(\mathsf{vk}, m)$: This oracle outputs a signature on m valid under the verification key vk. The requested message is added to the signed messages set $\mathcal{SM}$.

```
$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk}, \mathsf{OSign}(\mathsf{vk}, \cdot))$
4. $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5. IF $m^* \in \mathcal{SM}$ RETURN 0
6. ELSE RETURN $b$
```

The probability of success against this game is denoted by $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K}) = 1]$, $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(\mathfrak{K}, t) = \max_{\mathcal{A} \leq t} \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S},\mathcal{A}}(\mathfrak{K})$.

Encryption. An encryption scheme $\mathcal{C}$ is described by four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$):

– $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
– $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key pk and a (private) decryption key dk;
– $\mathsf{Encrypt}(\mathsf{ek}, M; \rho)$ outputs a ciphertext $\mathcal{C}$, on M, under the encryption key pk, with the randomness ρ;
– $\mathsf{Decrypt}(\mathsf{dk}, \mathcal{C})$ outputs the plaintext M, encrypted in the ciphertext $\mathcal{C}$ or $\perp$.

Such encryption scheme is required to have the following security properties:

– *Correctness*: For every pair of keys $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every messages M, and every random ρ, we should have $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, M; \rho)) = M$.
– *Indistinguishability under Adaptive Chosen Ciphertext Attack* $\mathsf{IND-CCA}$ ([NY90, RS92]):
  - $\mathsf{IND-CCA}$: An adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and ask several decryption of ciphertexts different from challenge one.
    The $\mathsf{ODecrypt}$ oracle outputs the decryption of c under the challenge decryption key dk. The input queries (c) are added to the list $\mathcal{CT}$ of decrypted ciphertexts.

```
$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{pk}, \mathsf{ODecrypt}(\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot))$
6. IF $(c^*) \in \mathcal{CT}$ RETURN 0
7. ELSE RETURN $b'$
```

Commitment. Commitments allow a user to commit to a value without revealing it, but without the possibility to later change his mind. It is composed of these algorithms:

– $\mathsf{SetupCom}(1^{\mathfrak{K}})$ generates the system parameters, according to the security parameter $\mathfrak{K}$.
– $\mathsf{KeyGen}(\mathsf{param})$ generates a commitment key ck, and possibly some verification key vk.
– $\mathsf{Commit}(\mathsf{ck}, \mathsf{vk}, m; r)$ produces a commitment c on the input message $m \in \mathcal{M}$ using the random coins $r \overset{\$}{\leftarrow} \mathcal{R}$.
– $\mathsf{Decommit}(\mathsf{ck}, c, m; r)$ opens the commitment c and reveals the message m.

Such a commitment scheme should be both *hiding*, which says that the commit phase does not leak any information about m, and *binding*, which says that the decommit phase should not be able to open to two different messages. Additional features are also sometimes required, such as non-malleability, extractability, and/or equivocability. We may also include a label ℓ, which is an additional public information that has to be the same in both the commit and the decommit phases.

A commitment scheme is said *equivocable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and two algorithms

– $\mathsf{SimCom}^{\ell}(\tau)$ that takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
– $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ that takes as input a commitment C, a label ℓ, a message x, an equivocation key eqk, and outputs an opening data δ for C and ℓ on x.

such as the following properties are satisfied: *trapdoor correctness* (all simulated commitments can be opened on any message), *setup indistinguishability* (one cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$) and *simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments), denoting by SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \overset{\$}{\leftarrow} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$.

A commitment scheme $\mathcal{C}$ is said *extractable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and a new algorithm

– $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

such as the following properties are satisfied: *trapdoor correctness* (all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$), *setup indistinguishability* (as above) and *binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x).

Chameleon Hash. This directly echoes to Chameleon Hashes, traditionally defined by three algorithms $\mathsf{CH} = (\mathsf{KeyGen}, \mathsf{CH}, \mathsf{Coll})$:
– $\mathsf{KeyGen}(\mathfrak{K})$: Outputs the chameleon hash key ck and the trapdoor tk;
– $\mathsf{CH}(\mathsf{ck}, m; r)$: Picks a random r, and outputs the chameleon hash a.
– $\mathsf{Coll}(\mathsf{ck}, m, r, m', \mathsf{tk})$: Takes as input the trapdoor tk, a start message and randomness pair (m, r) and a target message m' and outputs a target randomness r' such that $\mathsf{CH}(\mathsf{ck}, m; r) = \mathsf{CH}(\mathsf{ck}, m'; r')$.

The standard security notion for CH is collision resistance. Formally, CH is $(t, \varepsilon) - \mathsf{coll}$ if for the adversary $\mathcal{A}$ running in time at most t we have:

$$\Pr\left[\begin{array}{c}(\mathsf{ck}, \mathsf{tk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(\mathfrak{K}); ((m_1, r_1), (m_2, r_2)) \overset{\$}{\leftarrow} \mathcal{A}(\mathsf{ck}) \\ \wedge \quad \mathsf{CH}(\mathsf{ck}, m_1; r_1) = \mathsf{CH}(\mathsf{ck}, m_2; r_2) \wedge m_1 \neq m_2\end{array}\right] \leq \varepsilon.$$

However, any user in possession of the trapdoor tk is able to find a collision using Coll. Additionally, Chameleon Hash functions have the uniformity property, which means the hash value leaks nothing about the message input. Formally, for all pair of messages m_1 and m_2 and the randomly chosen r, the probability distributions of the random variables $\mathsf{CH}(\mathsf{ck}, m_1, r)$ and $\mathsf{CH}(\mathsf{ck}, m_2, r)$ are computationally indistinguishable.

We need here the hash value to be verifiable, so that we add two $\mathsf{VKeyGen}$ and Valid algorithms (executed by the receiver).

– $\mathsf{VKeyGen}(\mathsf{ck})$: Outputs the chameleon designated verification key vk by appending it to ck and the trapdoor vtk. This trapdoor can be empty or public if the chameleon hash is publicly verifiable.
– $\mathsf{Valid}(\mathsf{ck}, m, a, d, \mathsf{vtk})$: Allows to check that the sender knows how to open a Chameleon Hash a to a specific value m for the witness d. The verification can be public if vtk is empty or public, or specific to the receiver otherwise.

B Building Blocks

B.1 Classical Building Blocks

Waters Signature To sign scalar message in the standard model, one can use Waters Signatures [Wat05]. This signature scheme is defined by four algorithms:

Definition 15 (Waters Signature Scheme). $\mathcal{S} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*:*

– $\mathsf{Setup}(1^{\mathfrak{K}})$*, where* $\mathfrak{K}$ *is the security parameter, generates the global parameters* param *of the scheme, and more specifically the bilinear group* $(p, \mathbb{G}, \mathbb{G}_T, e, g)$*, an extra generator* h*, and generators* $(u_i)_{[\![0,k]\!]}$ *for the Waters function, where* k *is a polynomial in* $\mathfrak{K}$*,* $\mathcal{F}(m) = u_0 \prod_{i \in [\![1,k]\!]} u_i^{m_i}$*, where* $m = (m_1, \ldots, m_k) \in \{0,1\}^k$*.*
– $\mathsf{KeyGen}(\mathsf{param})$ *picks a random* $x \overset{\$}{\leftarrow} \mathbb{Z}_p$ *and outputs the secret key* $\mathsf{sk} = Y = h^x$*, and the verification key* $\mathsf{vk} = X = g^x$*;*
– $\mathsf{Sign}(\mathsf{sk}, m; \mu)$ *outputs a signature* $\sigma(m) = (Y\mathcal{F}(m)^\mu, g^{-\mu})$*;*
– $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ *checks the validity of* σ*, by checking if the following pairing equation holds:* $e(g, \sigma_1) \cdot e(\mathcal{F}(m), \sigma_2) \overset{?}{=} e(X, h)$

Theorem 16. *This scheme is* $\mathsf{EUF-CMA}$ *under the* CDH *assumption.*

Theorem 17. *Waters Signature is randomizable if we define:*

– $\mathsf{Random}(\mathsf{vk}, \mathcal{F}(m), \sigma = (\sigma_1, \sigma_2); \mu')$ *outputs* $\sigma' = (\sigma_1 \cdot \mathcal{F}(m)^{\mu'}, \sigma_2 \cdot g^{-\mu'})$*.*

Proof. We simply have:

$$\sigma = \mathsf{Sign}(\mathsf{sk}, \mathcal{F}(m); \mu) \quad \Rightarrow \quad \mathsf{Random}(\mathsf{vk}, \mathcal{F}(m), \sigma; \mu') = \mathsf{Sign}(\mathsf{sk}, \mathcal{F}(m); \mu + \mu' \mod p).$$

Due to the additive law in $\mathbb{Z}_p$, fresh signature distribution is indistinguishable from randomized one. □

CDH-based Chameleon Hash [BC15]

– $\mathsf{KeyGen}(\mathfrak{K})$: Outputs the chameleon hash key $\mathsf{ck} = (g, h)$ and the trapdoor $\mathsf{tk} = \alpha$, where $g^\alpha = h$;
– $\mathsf{VKeyGen}(\mathsf{ck})$: Appends f to ck and $\mathsf{vtk} = \log_g(f)$
– $\mathsf{CH}(\mathsf{ck}, m; r)$: Picks a random $r \in \mathbb{Z}_p$, and outputs the chameleon hash $a = h^r g^m$. Sets $d = f^r$.
– $\mathsf{Coll}(m, r, m', \mathsf{tk})$: outputs $r' = r + (m - m')/\alpha$.
– $\mathsf{Valid}(\mathsf{ck}, m, a, d, \mathsf{vtk})$: One can check if $a = h^m \cdot d^{1/\mathsf{vtk}}$.

In a pairing environment, there is a trivial way to check this CH using a pairing instead of knowing vtk.

This is a CDH variant of the Pedersen chameleon hash [Ped92].

ElGamal Encryption ElGamal encryption [ElG84] is defined by the following four algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$: The scheme needs a multiplicative group $(p, \mathbb{G}, g)$,. The global parameters param consist of these elements $(p, \mathbb{G}, g)$.
- $\mathsf{KeyGen}(\mathsf{param})$: Chooses one random scalar $\mu \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, which define the secret key $\mathsf{dk} = \mu$, and the public key $\mathsf{ek} = X = g^{\mu}$.
- $\mathsf{Encrypt}(\mathsf{ek} = X, M; \alpha)$: For a message $M \in \mathbb{G}$ and a random scalar $\alpha \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, computes the ciphertext as $\boldsymbol{c} = \big(c_1 = X^{\alpha} M, c_2 = g^{\alpha}\big)$.
- $\mathsf{Decrypt}(\mathsf{dk} = \mu, \boldsymbol{c} = (c_1, c_2))$: One computes $M = c_1/(c_2^{\mu})$.

As shown by Boneh [Bon98], this scheme is $\mathsf{IND\text{-}CPA}$ under the hardness of DDH.

Cramer-Shoup Encryption The Cramer-Shoup encryption scheme [CS98] is an $\mathsf{IND\text{-}CCA}$ version of the ElGamal Encryption. We present it here as a labeled public-key encryption scheme, the classical version is done with $\ell = \emptyset$.

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \stackrel{\$}{\leftarrow} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^{z}$. It also chooses a Collision-Resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^{\xi})^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/(u_1^{z})$ and outputs M. Otherwise, one outputs $\perp$.

The security of the scheme is proven under the DDH assumption and the fact the hash function used is a Universal One-Way Hash Function. A generalization of this encryption to the $k - \mathsf{MDDH}$ assumption can be found further below.

Smooth Projective Hash Functions on Cramer Shoup Encryption
One can now build a Hash Proof system on this CCA-2 scheme:

- $\mathsf{HashKG}(\mathfrak{L}_M) : \mathsf{hk} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^4$,
- $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2))$: Setting $\mathsf{ht} = (h, g_1, g_2, cd^{\theta})$, we have $\mathsf{hp} = h^{\mathsf{hk}_1} g_1^{\mathsf{hk}_2} g_2^{\mathsf{hk}_3} \left(cd^{\theta}\right)^{\mathsf{hk}_4}$, where $\theta = H(\ell, \boldsymbol{e})$
- $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2)) : H \leftarrow (\mathcal{C}_1/M)^{\mathsf{hk}_1} \mathcal{C}_2^{\mathsf{hk}_2} \mathcal{C}_3^{\mathsf{hk}_3} \mathcal{C}_4^{\mathsf{hk}_4}$,
- $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2), r) H' \leftarrow \mathsf{hp}^r$

B.2 *k*-MDDH Building Blocks

In this section we extend classical building blocks to the MDDH assumptions. While most of them were at least implicitly defined before in [EHK+13], we feel it may be useful to group them together for ease of reading.

Chameleon Hash We first extend the Pedersen commitment, to obtain a compatible verifiable Chameleon Hash functions:

- $\mathsf{KeyGen}(\mathfrak{K})$: Outputs the chameleon hash key $\mathsf{ck}_1 = \boldsymbol{F} \stackrel{\$}{\leftarrow} \mathcal{D}_k$ and the trapdoor $\mathsf{tk} = \underline{\boldsymbol{F}} \cdot \overline{\boldsymbol{F}^{-1}}$, it also generates $\boldsymbol{G} \stackrel{\$}{\leftarrow} \mathsf{GL}_k$ and adds $\mathsf{ck}_2 = [\boldsymbol{E}\boldsymbol{G}]_2$ to the key ck and keeps the verification trap $\mathsf{vtk} = \boldsymbol{G}^{-1}$
- $\mathsf{CH}(\mathsf{ck}, m; \boldsymbol{\rho})$: Picks a random $\boldsymbol{\rho} \in \mathbb{Z}_p^{k+1}$, and outputs the chameleon hash $[\boldsymbol{a}]_2 = [(\boldsymbol{\rho}^{\top} \mid m)\mathsf{ck}_1]_2$. Sets $\boldsymbol{d} = \boldsymbol{\rho}^{\top}\mathsf{ck}_2$.
- $\mathsf{Coll}(m, \boldsymbol{\rho}, m', \mathsf{tk})$: outputs $\boldsymbol{\rho'} = \boldsymbol{\rho} + (m - m')\mathsf{tk}$.

– $\mathsf{Valid}(\mathsf{ck}, m, [\boldsymbol{a}]_2, [\boldsymbol{d}]_2, \mathsf{vtk})$: The user outputs $[\boldsymbol{d}]_2$, so that one can check if $\boldsymbol{a} = \boldsymbol{d}^\top \mathsf{vtk} + \boldsymbol{m}\underline{\boldsymbol{F}}$.

Correctness follows easily, finding a collision leads to directly to computing tk, and so $\boldsymbol{G}$ from $[\boldsymbol{E}]_2$ and vtk. Such Pedersen Commitment was already used in the master public key generation in [BKP14]. It also corresponds to the $k-\mathsf{MDDH}$ version of the Haralambiev [Har11, Section 4.1.4] TC4 commitment scheme, called TC4 which was revisited in recent works [ABB+13, BC15] as the basis for a UC secure commitment.

k-MDDH Linear Encryption [EHK+13] provides a CPA encryption scheme:

– $\mathsf{KeyGen}(\mathfrak{K})$: Picks $\boldsymbol{E} \xleftarrow{\$} \mathcal{D}_k$, sets $\mathsf{pk} = [\boldsymbol{E}]_2, \mathsf{sk} = (\underline{\boldsymbol{E}} \cdot \overline{\boldsymbol{E}^{-1}})$ to be the public encryption key.
– $\mathsf{Encrypt}(\mathsf{ek}, M, \ell; \boldsymbol{\mu})$ If $M \in \mathbb{G}_2$, $\boldsymbol{e} = \mathsf{pk}\boldsymbol{\mu} + \begin{bmatrix} 0 \\ M \end{bmatrix}_2$.
– $\mathsf{Decrypt}(\mathsf{dk}, \boldsymbol{e}, \ell)$ Outputs $M = \mathsf{sk}\overline{\boldsymbol{e}} + \underline{\boldsymbol{e}}$.

$2m$ labelled k-MDDH Multi Cramer Shoup Encryption We are going to need a CCA-2 encryption, SPHF friendly, and proven under MDDH. Fortunately for us, [EHK+13] also provides a compatible Universal$_2$ Hash Proof system, which thanks to [CS02] leads to the required scheme:

– $\mathsf{KeyGen}(\mathfrak{K})$: Picks $\boldsymbol{E} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{u}, \boldsymbol{v} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, sets $\mathsf{pk} = ([\boldsymbol{E}]_2, [\boldsymbol{u}^\top \boldsymbol{E}]_2, [\boldsymbol{v}^\top \boldsymbol{E}]_2), \mathsf{sk} = (\underline{\boldsymbol{E}} \cdot \overline{\boldsymbol{E}^{-1}}, \boldsymbol{u}, \boldsymbol{v})$ to be the public encryption key, where $\mathcal{H}$ is a random collision-resistant hash function from $\mathcal{H}$ [13].
– $\mathsf{Encrypt}(\mathsf{ek}, M, \ell; \boldsymbol{\mu})$ If $M \in \mathbb{G}_2$, $\mathcal{C} = (\boldsymbol{e} = \mathsf{pk}_1\boldsymbol{\mu} + \begin{bmatrix} 0 \\ M \end{bmatrix}_2, w = [(\mathsf{pk}_2 + \theta \mathsf{pk}_3)\boldsymbol{\mu}]_2$. where $\theta = H(\ell, \boldsymbol{e})$.
– $\mathsf{Decrypt}(\mathsf{dk}, \mathcal{C}, \ell)$ If $[w]_2 \stackrel{?}{=} [(\boldsymbol{u}^\top + \theta \boldsymbol{v}^\top)\overline{\boldsymbol{e}}]_2$, then outputs $M = \mathsf{sk}\overline{\boldsymbol{e}} + \underline{\boldsymbol{e}}$.

The above scheme can be extended naturally to encrypt matrices of group elements $\boldsymbol{D} = (\boldsymbol{D}_1, \ldots, \boldsymbol{D}_{2m}) \in \mathbb{G}_2^{2m}$, by having $2m$ tuples of random scalars in the secret key, and a global value θ for the encryption. Following the techniques from [ABB+13], and the work from [EHK+13] this scheme is `VIND-PO-CCA` under the MDDH assumption.

k-linear Smooth Projective Hash Function

One can now build a Hash Proof system on this CCA-2 scheme, by following [EHK+13, BBC+13b]

– $\mathsf{HashKG}(\mathfrak{L}_M) : \mathsf{hk} \xleftarrow{\$} \mathbb{Z}_p^{k+2}$,
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2))$: Setting $\mathsf{ht} = \dfrac{\mathsf{pk}_1}{\mathsf{pk}_2 + \theta \mathsf{pk}_3}$, we have $[\mathsf{hp}]_2 = [\mathsf{hk}^\top \mathsf{ht}]_2$, where $\theta = H(\ell, \boldsymbol{e})$,
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2)) : [H]_2 \leftarrow [\mathsf{hk}^\top (\mathcal{C} - (0 \mid M)^\top)]_2$,
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}_M, \ell, [\mathcal{C}]_2), \boldsymbol{\mu})[H']_2 \leftarrow [\mathsf{hp}\boldsymbol{\mu}]_2$

The smoothness of such system is show by considering the determinant of the matrix $\boldsymbol{D} = \begin{bmatrix} \mathsf{ht}^\top \\ \boldsymbol{C} - (0 \mid M)^\top \end{bmatrix}$

As soon as $\boldsymbol{C}$ is not a valid encryption of M, this matrix has non zero determinant. As the first lines leads to $\mathsf{hk}^\top \mathsf{ht}$ the public projection key hp, and the last to $\mathsf{hk}^\top(\boldsymbol{C} - (0 \mid M))$ the computed Hash value, we can deduce that form the public views, the computed hash value is information theoretically independent from the projection keys for words outside the language.

[13] Like Cramer-Shoup one could rely on an universal one-way hash function family instead

C Proof of the Security of Fragmented IBE

In this section, we briefly prove that the variation over the [BKP14] IBE, does not weaken it's security.

Theorem 18. *The Blinded IBE achieves the leak-free secret key generation requirements under the security of the initial IBE, the extractability of the* MDDH *Cramer Shoup Encryption and the Smoothness of the SPHF on the bit commitment.*

Proof. Given an adversary $\mathcal{A}$ against the security of the blinded scheme, we are going to build an adversary $\mathcal{B}$ against the security of the initial IBE.

From the challenge, $\mathcal{B}$ receives the parameter from an IBE scheme mpk, has access to a user key generation oracle $\mathsf{USKGen}_{\mathcal{O}}$, and for a given fresh id^*, a tuple $(\mathsf{K}^*, \mathsf{C}^*)$ whose consistency he need to decide.

In the initial game $\mathcal{G}_0$, $\mathcal{B}$ behaves normally, generating $\mathsf{mpk}, \mathsf{msk}$, and answering $\mathcal{A}$ blinded request honestly.

$\mathcal{G}_1$ In this game, $\mathcal{B}$ starts altering his answer. Using the extraction procedure on the CCA commitment, $\mathcal{B}$ is able to recover the identity id queried by $\mathcal{A}$. If for a bit i, there is two valid openings, then $\mathcal{A}$ broke the collision resistance property of the underlying chameleon hash (and so MDDH) and $\mathcal{B}$ aborts. The Chameleon hash being Collision Resistant, this game is equivalent to the previous one under $k - \mathsf{MDDH}$.

$\mathcal{G}_2$ In this game, $\mathcal{B}$ starts altering his answer. Using the extraction procedure on the CCA commitment, $\mathcal{B}$ is able to recover the identity id queried by $\mathcal{A}$. Now, for each bit, there is at least one dummy value $\bar{\mathsf{id}}_i$, and so $\mathcal{B}$ computes random values $\boldsymbol{\omega}_{i,\bar{\mathsf{id}}_i} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$. Under the smoothness of the underlying smooth projective hash function, this game is indistinguishable from the previous one.

$\mathcal{G}_3$ Now $\mathcal{B}$ continues to extract the requested id, picks random $\boldsymbol{f} \xleftarrow{\$} \mathbb{Z}_p^{\ell \times k+1}$, and sets $\boldsymbol{w}_0 = \mathsf{usk}[\mathsf{id}] - \sum_{i=1}^{\ell} \boldsymbol{f}_i$, then for $i \in [\![1, n]\!]$, $\boldsymbol{\omega}_{i,\mathsf{id}_i} = \boldsymbol{f}_i + H_{i,\mathsf{id}_i}$ while $\boldsymbol{\omega} i, \mathsf{id}_i \xleftarrow{\$} \mathbb{Z}_p^{k+1}$ as before. If $\mathcal{B}$, does not recover a valid identity, he simply only sends dummy values.
This game is indistinguishable from the previous one, as this is just a rewriting of the vector $\boldsymbol{f}_i$. (Noting $\hat{\boldsymbol{f}}_i$ the old one, we have $\boldsymbol{f}_i = \hat{\boldsymbol{f}}_i - (\mathsf{id}_i \boldsymbol{Y}_i)\boldsymbol{t}$ which follows the same distribution).

$\mathcal{G}_4$ $\mathcal{B}$ can now forget about msk, when receiving a $\mathsf{BlindUSKGen}$ request, $\mathcal{B}$ extracts the request identity, and if it is not $\perp$ he forwards it to the $\mathsf{USKGen}_{\mathcal{O}}$ oracle, and plugs the received $\mathsf{usk}[\mathsf{id}]$ as before.

Now $\mathcal{A}$ can request a challenge from $\mathcal{B}$, $\mathcal{B}$ forwards this request to the initial non-blinded IBE challenge for a fresh id^*, and returns the challenge $(\mathsf{K}^*, \mathsf{C}^*)$ to $\mathcal{A}$ which leads to the conclusion. □

D Proof of the Generic Construction of Adaptive Oblivious Transfer

We prove the security of this protocol via a sequence of games, starting from the real game, where the adversary $\mathscr{A}$ interacts with the real players, and ending with the ideal game, where we have built a simulator $\mathscr{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathscr{A}$. The simulator is explicitely given in Game 11. Recall that we consider adaptive corruptions.

We denote as $\mathcal{S}$ the server and $\mathcal{U}$ the user. The main idea is that, by assumption, the simulator can always obtain the common trapdoor tk of the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ and use it to commit to a word simultaneously belonging to all the languages. In case of adaptive corruption, we face two cases: either the word was not correct, in which case, following the real protocol, the user should have erased his randomness, so that the simulator does not have anything to reveal. Or the word was correct and should belong to a certain language $\mathfrak{L}_s$, and the simulator can then adapt the word and randomness so that it seems to belong to $\mathfrak{L}_s$ (only). This enables us to avoid the use of commitments both extractable and equivocable (which is the usual tool for adaptive corruptions).

Due to the construction of the protocol, we have to prove that the user recovers the secret key $\mathsf{usk}[s]$ corresponding to the s-th line of the database in an oblivious way, which means on the one hand that the user gains no information on the other keys, and on the other hand that the server gains no information on the key required by the user. Assuming this is the case (the proof follows), the adaptive security of the global oblivious transfer relies on the security of the underlying IBE scheme: The indistinguishability of the ciphertexts ensure that the user only recovers the s-th line for which he knows the secret key $\mathsf{usk}[s]$.

Since the channels are authenticated, we know whether a flow was sent by an honest player (and received without any alteration) or not.

Game G_0: This is the real game.

Game G_1: In this game, the simulator generates correctly every flow on behalf of the honest players, as they would do themselves, knowing the database $(DB(1), \dots, DB(n))$ and the word W sent by the environment to the server and the user. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{U})$. In case of corruption of an honest player (either server or user), the simulator can give the internal data generated on behalf of the honest users.

Game G_2: In this game, we replace the setup algorithm Setup by SetupT, allowing the existence of a trapdoor to find words in the intersection of the collection of languages. We also allow the simulator to program $\mathsf{Setup}_{\mathsf{CCA}}$ in the CRS, enabling it to learn the extraction trapdoor of the CCA encryption scheme. The indistinguishability of the setups makes this game indistinguishable from the former one for the environment. Corruptions are handled the same way.

Game G_3: We first deal with honest servers $\mathcal{S}$: he computes everything honestly during the database preparation. When receiving a commitment C, the simulator extracts the committed value W. By testing with the help of the algorithm Verify, it recovers s such that $W \in \mathfrak{L}_s$. If it recovers $s \neq t$ such that $W \in \mathfrak{L}_s \cap \mathfrak{L}_t$, then the adversary has broken the infeasibility of finding a word in an intersection of languages without knowing the trapdoor and we abort the game. Otherwise, instead of computing the keys $H_{i,b}$, for $i = 1, \dots, \ell$ and $b = 0, 1$ with the hash function, the simulator then chooses $H_{i,b} \overset{\$}{\leftarrow} G$ when b is not equal to the i-th bit of W.

With an hybrid proof, applying the smoothness for every honest sender, on every index (i, b) such that $b \neq W_i$, since C is extracted to $W \in \mathfrak{L}_i$, for any (i, b) such that $b \neq W_i$, the hash value is indistinguishable from a random value.

In case of corruption, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly). This game is thus indistinguishable from the previous one under the smoothness.

Game G_4: Still in this case, when receiving a commitment C, the simulator extracts the committed value W. By testing with the help of the algorithm Verify, it recovers s such that $W \in \mathfrak{L}_s$. Instead of proceeding as the server would do with $(\mathsf{usk}[i, b])$, the simulator proceeds on $(\mathsf{usk}'[i, b])$, with $\mathsf{usk}'[i, b] = 0$ except if $b = W_i$. Since the masks $H_{i,b}$, for $b \neq W_i$, are random, this game is perfectly indistinguishable from the previous one.

Game G_5: We now deal with honest users $\mathcal{U}$: the simulator now uses the trapdoor tk to find a word W' in the intersection of all languages.

With an hybrid proof, applying a security game in each session in which the simulator does not know the trapdoor tk, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value W, thus s.

Game $\boldsymbol{G}_6$: We deal with **the generation of R for honest servers $\mathcal{S}$ where the users $\mathcal{U}$ are honests**: if $\mathcal{S}$ and $\mathcal{U}$ are honest at least until $\mathcal{S}$ received the second flow, the simulator sets $R = F(S')$ for both $\mathcal{S}$ and $\mathcal{U}$, with S' a random value, instead of $R = F(S)$.

With an hybrid proof, applying the $\mathsf{IND\text{-}CPA}$ property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_7$: Still in the same case, the simulator sets R as a random value, instead of $R = F(S')$.

With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_8$: We now deal with **the generation of H_{i,W_i} for honest servers $\mathcal{S}$ with honest users $\mathcal{U}$**. Thanks to the additional random mask R, one can send random $(\mathsf{usk}[i, W_i])_i$, and H_{i,W_i} can be computed later (when $\mathcal{U}$ actually receives its flow).

As above, but only if $\mathcal{U}$ has not been corrupted before receiving its flow, the simulator chooses $H_{i,W_i} \overset{\$}{\leftarrow} G$. With an hybrid proof, applying the pseudo-randomness, for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitment information for C. If the player $\mathcal{U}$ involved in the pseudo-randomness game gets corrupted (but the decommitment information is unknown) we are not in this case, and we can thus abort it.

In case of corruption of $\mathcal{S}$, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly).

In case of corruption of the receiver $\mathcal{U}$, and thus receiving the value $\widetilde{DB(s)}$, the simulator computes $\tilde{K}_s$ such that $\widetilde{K_s} \oplus \widetilde{DB(s)} = K_s \oplus DB(s)$ and the corresponding $\widetilde{\mathsf{usk}[W]}$. It chooses R (because it was a random value unknown to the adversary and all the other $H_{i,b}$ are independent random values too) such that $\left(\bigoplus_i \mathsf{busk}[i, W_i] \oplus H'_{i,W_i}\right) \oplus Z \oplus R = \widetilde{\mathsf{usk}[W]}$.

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Remark. We now explain how, in the pairing instantiation of our protocol, given already sent values $D_s, K_s \oplus DB(s)$, a simulator recovering from the environment the value $\widetilde{DB(s)}$, can adaptively be able to change his memory so as to compute a user key $\widetilde{\mathsf{usk}[W]}$ such that $\mathsf{Dec}(\widetilde{\mathsf{usk}[W]}, W, D_s) = DB(s) \oplus K_s \oplus \widetilde{DB(s)}$. This is exactly where we use the restriction on the size of DB elements so that we can manage to find a vector $\delta_s \in \mathbb{G}_2^{2k+1}$ such that $DB(s) \oplus \widetilde{DB(s)} = e(D_s, \delta_s)$. Thus, this allows the server to update his memory into $\widetilde{\mathsf{usk}[W]} = \mathsf{usk}[W] \cdot \delta_s$.

Game $\boldsymbol{G}_9$: Still in this case, the simulator proceeds on $(\mathsf{usk}[i, b])$, with $\mathsf{usk}[i, b] = 0$ for all i, b. Since the masks $H_{i,b}$, $Z \oplus R$, for any i, b, are independent random values (the $\mathsf{busk}[i, b]$, for $b \neq W_i$ are independent random values, and R is independently random), this game is perfectly indistinguishable from the previous one.

We remark that it is therefore no more necessary to know the index s given by the ideal functionality to the honest receiver $\mathcal{U}$, to simulate $\mathcal{S}$ (but it is still necessary to simulate $\mathcal{U}$).

Game $\boldsymbol{G}_{10}$: We do not use anymore the knowledge of s when simulating an **honest user $\mathcal{U}$**: the simulator generates a word W' in the intersection of the languages and $C \overset{\$}{\leftarrow} \mathsf{Encrypt}^{\ell}_{\mathsf{cca}}(W'; \boldsymbol{\rho})$, with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R})$, to send C during the first phase of honest users. This does not change anything from the previous game since the randomness needed to open to a word in another language is never revealed.

When it thereafter receives $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R}, (\mathsf{hp}_{i,b}, \mathsf{busk}[i, b]))$ from the adversary, the simulator computes, for all lines t, $\mathsf{usk}[W_t]$ and recovers K_t and finally $DB(t)$, which provides the database $(DB(1), \ldots, DB(n))$ submitted by the sender. It uses them to send a $\mathtt{Send}$-message to the ideal functionality.

Game $\boldsymbol{G}_{11}$: We can now make use of the functionality, which leads to the following simulator:

- when receiving a `Send`-message from the ideal functionality, which means that an honest server has sent a pre-flow and a database, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \stackrel{\$}{\leftarrow} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted sender) and a `Receive`-message from the ideal functionality, which means that an honest receiver has sent a first flow, the simulator generates a word W' in the intersection of languages, $C \stackrel{\$}{\leftarrow} \mathsf{Encrypt}^{\ell}_{\mathsf{cca}}(W; \boldsymbol{\rho})$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$ and $c \stackrel{\$}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}, S)$ where S is a random value;
- when receiving a commitment C and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator extracts the committed value W and recovers s (aborting in case of multiple values), and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as S, and computes $R = F(S)$);
- when receiving $(\mathsf{hp}_{i,b}, \mathsf{busk}[i, b])$ from the adversary, the simulator computes, for all lines t, $\mathsf{usk}[W_t]$ and recovers K_t and finally $DB(t)$, which provides the database $(DB(1), \ldots, DB(n))$ submitted by the sender. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with $\widetilde{DB(s)}$, on behalf of a corrupted receiver, from the extracted W leading to s, instead of proceeding as the sender would do on $(\mathsf{usk}[i, b])$, the simulator proceeds on $(\mathsf{usk}'_{[}i, b])$, with $\mathsf{usk}'_{[}i, b] = 0$ except if $b = W_i$.
- when receiving a commitment C and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(\mathsf{usk}'_{[}i, b])$, with $\mathsf{usk}'_{[}i, b] = 0$ except if $b = W_i$, but it chooses R uniformly at random instead of choosing it as $R = F(S)$; in case of corruption afterward, the simulator will adapt R such that $\left(\bigoplus_i \mathsf{busk}[i, W_i] \oplus H'_{i,W_i}\right) \oplus Z \oplus R = \widetilde{\mathsf{usk}[W]}$, where $\widetilde{\mathsf{usk}[W]}$ leads to $\tilde{K}_s$ such that $\widetilde{K_s} \oplus \widetilde{DB(s)} = K_s \oplus DB(s)$, where $\widetilde{DB(s)}$ is the message actually received by the receiver.

Any corruption either reveals s earlier, which allows a correct simulation of the receiver, or reveals $(DB(1), \ldots, DB(n))$ earlier, which allows a correct simulation of the server. When the server has sent his flow, he has already erased all his random coins.

However, there would have been an issue when the user is corrupted after the server has sent is flow, but before the user receives it, since he has kept $\boldsymbol{\rho}$: this would enable the adversary to recover $\mathsf{usk}[W]$ from $\mathsf{busk}[i, W_i]$ and hp_{i,W_i}. This is the goal of the ephemeral mask R that provides a secure channel.

E Proof of the Generic Construction of Oblivious Language-Based Envelope

We prove the adaptive[14] security of this protocol via a sequence of games, starting from the real game, where the adversary $\mathscr{A}$ interacts with the real players, and ending with the ideal game, where we have built a simulator $\mathscr{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathscr{A}$.

We denote as $\mathcal{S}$ the sender (*i.e.* the server) and $\mathcal{R}$ the receiver (*i.e.* the user). The main idea is that, by assumption, the simulator can always obtain the common trapdoor tk of the collection of languages $(\mathfrak{L}_1, \ldots, \mathfrak{L}_n)$ and use it to commit to a word simultaneously belonging to all the languages. In case of adaptive corruption, we face two cases: either the word was not correct, in which case, following the real protocol, the user should have erased his randomness, so that the simulator does not have anything to reveal. Or the word was correct and should belong to a certain language $\mathfrak{L}_i$, and the simulator can then adapt the word and randomness so that it seems to belong to $\mathfrak{L}_i$ (only). This enables us to avoid the use of commitments both extractable and equivocable (which is the usual tool for adaptive corruptions).

[14] One can obtain the proof of the static version by removing the parts related to the pre-flow and to the adaptive corruptions in the proof below.

We say that a flow is *oracle-generated* if it was sent by an honest player (or the simulator) and received without any alteration by the adversary. It is said *non-oracle-generated* otherwise.

Game $\boldsymbol{G}_1$: This is the real game.

Game $\boldsymbol{G}_2$: In this game, the simulator generates correctly every flow on behalf of the honest players, as they would do themselves, knowing the inputs $(m_1, \ldots, m_n)$ and W sent by the environment to the sender and the receiver. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R})$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\boldsymbol{G}_3$: In this game, we replace the setup algorithm Setup by SetupT, allowing the existence of a trapdoor to find words in the intersection of the collection of languages. We also allow the simulator to program $\mathsf{Setup}_{\mathsf{CCA}}$ in the CRS, enabling it to learn the extraction trapdoor of the CCA encryption scheme. The indistinguishability of the setup makes this game indistinguishable from the former one for the environment. Corruptions are handled the same way.

Game $\boldsymbol{G}_4$: We first deal with oracle-generated flows from the senders $\mathcal{S}$: when receiving a commitment C, the simulator extracts the committed value W. By testing with the help of the algorithm Verify, it recovers i such that $W \in \mathfrak{L}_i$. If it recovers $i \neq j$ such that $W \in \mathfrak{L}_i \cap \mathfrak{L}_j$, then the adversary has broken the infeasibility of finding a word in an intersection of languages without knowing the trapdoor and we abort the game. Otherwise, instead of computing the key v_t, for $t = 1, \ldots, n$ with the hash function, the simulator then chooses $v_t \xleftarrow{\$} G$ for $t \neq i$.

With an hybrid proof, applying the smoothness for every honest sender, on every index $t \neq i$, since C is extracted to $W \in \mathfrak{L}_i$, for any $t \neq i$, the hash value is indistinguishable from a random value.

In case of corruption, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly). This game is thus indistinguishable from the previous one under the smoothness.

Game $\boldsymbol{G}_5$: Still in this case, when receiving a commitment C, the simulator extracts the committed value W, giving it the number i. Instead of proceeding as the sender would do on $(m_1, \ldots, m_n)$, the simulator proceeds on $(m'_1, \ldots, m'_n)$, with $m'_i = m_i$, but $m'_t = 0$ for all $t \neq i$. Since the masks v_t, for $t \neq i$, are random, this game is perfectly indistinguishable from the previous one.

Game $\boldsymbol{G}_6$: We now deal with oracle-generated flows from the receivers $\mathcal{R}$: the simulator now uses the trapdoor tk to find a word W in the intersection of all languages.

With an hybrid proof, applying a security game in each session in which the simulator does not know the trapdoor tk, one can show the indistinguishability of the two games. In case of corruption of the receiver, one learns the already known value i.

Game $\boldsymbol{G}_7$: We deal with **the generation of R for honest senders $\mathcal{S}$ on oracle-generated queries (adaptive case only)**: if $\mathcal{S}$ and $\mathcal{R}$ are honest at least until $\mathcal{S}$ received the second flow, the simulator sets $R = F(J')$ for both $\mathcal{S}$ and $\mathcal{R}$, with J' a random value, instead of $R = F(J)$.

With an hybrid proof, applying the $\mathsf{IND\text{-}CPA}$ property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_8$: Still in the same case, the simulator sets R as a random value, instead of $R = F(J')$.

With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_9$: We now deal with **the generation of v_i for honest senders $\mathcal{S}$ on oracle-generated queries**:

- in the static case (the pre-flow is only needed to compute $(\mathsf{vk}, \mathsf{vtk})$, and thus we assume $R = 0$) the simulator chooses $v_i \overset{\$}{\leftarrow} G$ (for $t \neq i$, the simulator already chooses $v_t \overset{\$}{\leftarrow} G$), where i is the index given by the ideal functionality to the honest receiver $\mathcal{R}$.
  With an hybrid proof, applying the pseudo-randomness for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitment information for C;
- in the adaptive case, and thus with the additional random mask R, one can send a random m_i, and v_i can be computed later (when $\mathcal{R}$ actually receives its flow).
  As above, but only if $\mathcal{R}$ has not been corrupted before receiving its flow, the simulator chooses $v_s \overset{\$}{\leftarrow} G$. With an hybrid proof, applying the pseudo-randomness, for every honest sender, the hash value is indistinguishable from a random value, because the adversary does not know any decommitment information for C. If the player $\mathcal{R}$ involved in the pseudo-randomness game gets corrupted (but the decommitment information is unknown) we are not in this case, and we can thus abort it.
  In case of corruption of $\mathcal{S}$, everything has been erased (except after the pre-flow, where the simulator can reveal the keys $(\mathsf{pk}, \mathsf{sk})$ generated honestly).
  In case of corruption of the receiver $\mathcal{R}$, and thus receiving the value m_i, the simulator chooses R (because it was a random value unknown to the adversary and all the other v_t are independent random values too) such that

$$R \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (L_i, \mathsf{param}), (\ell, C), r_i) \oplus m_i = Q_i.$$

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Game G_{10}: Still in this case, the simulator proceeds on $(m'_1, \ldots, m'_n)$, with $m'_t = 0$ for all i. Since the masks $v_t \oplus R$, for any $t = 1, \ldots, n$, are independent random values (the v_t, for $t \neq i$ are independent random values, and v_i is also independently random in the static case, while R is independently random in the adaptive case), this game is perfectly indistinguishable from the previous one.

We remark that it is therefore no more necessary to know the index i given by the ideal functionality to the honest receiver $\mathcal{R}$, to simulate $\mathcal{S}$ (but it is still necessary to simulate $\mathcal{R}$).

Game G_{11}: We do not use anymore the knowledge of i when simulating an **honest receiver** $\mathcal{R}$: the simulator generates a word W in the intersection of the languages and $C \overset{\$}{\leftarrow} \mathsf{Encrypt}^{\ell}_{\mathsf{cca}}(W; r)$, with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R})$, to send C during the first phase of honest receivers. This does not change anything from the previous game since the randomness needed to open to a word in another language is never revealed.

When it thereafter receives $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathcal{S}, \mathcal{R}, (Q_1, \mathsf{hp}_1, \ldots, Q_n, \mathsf{hp}_n))$ from the adversary, the simulator computes, for $i = 1, \ldots, k$, r_i,

$$v_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_i, \mathsf{param}), (\ell, C), r_i)$$

and $m_i = v_i \oplus R \oplus Q_i$. This provides the database submitted by the sender.

Game G_{12}: We can now make use of the functionality, which leads to the following simulator:

- when receiving a $\mathtt{Send}$-message from the ideal functionality, which means that an honest sender has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted sender) and a $\mathtt{Receive}$-message from the ideal functionality, which means that an honest receiver has sent a first flow, the simulator generates a word W in the intersection of languages, $C \overset{\$}{\leftarrow} \mathsf{Encrypt}^{\ell}_{\mathsf{cca}}(W; r)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, \mathcal{R}, \mathcal{S})$ and $c \overset{\$}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}, J)$ where R is a random value;

- when receiving a commitment C and a ciphertext c, generated by the adversary (from a corrupted receiver), the simulator extracts the committed value W and recovers i (aborting in case of multiple values), and uses it to send a `Receive`-message to the ideal functionality (and also decrypts the ciphertext c as J, and computes $R = F(J)$);
- when receiving $(Q_1, \mathsf{hp}_1, \ldots, Q_n, \mathsf{hp}_n)$ from the adversary (a corrupted sender), the simulator computes, for $i = 1, \ldots, n$, r_i,

$$v_i \leftarrow \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_i, \mathsf{param}), (\ell, C), r_i)$$

  and $m_i = v_i \oplus R \oplus Q_i$. It uses them to send a `Send`-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with m_i, on behalf of a corrupted receiver, from the extracted W leading to i, instead of proceeding as the sender would do on $(m_1, \ldots, m_n)$, the simulator proceeds on $(m'_1, \ldots, m'_n)$, with $m'_i = m_i$, but $m'_j = 0$ for all $j \neq i$;
- when receiving a commitment C and a ciphertext c, generated by an honest sender (*i.e.*, by the simulator itself), the simulator proceeds as above on $(m'_1, \ldots, m'_n)$, with $m'_j = 0$ for all j, but it chooses R uniformly at random instead of choosing it as $R = F(J)$; in case of corruption afterward, the simulator will adapt R such that $R \oplus \mathsf{ProjHash}(\mathsf{hp}_i, (\mathfrak{L}_i, \mathsf{param}), (\ell, C), r_i) \oplus Q_i = m_i$, where m_i is the message actually received by the receiver.

Any corruption either reveals i earlier, which allows a correct simulation of the receiver, or reveals $(m_1, \ldots, m_n)$ earlier, which allows a correct simulation of the sender. When the sender has sent his flow, he has already erased all his random coins.

However, there would have been an issue when the receiver is corrupted after the sender has sent is flow, but before the receiver receives it, since he has kept r_i: this would enable the adversary to recover m_i from Q_i and hp_i. This is the goal of the ephemeral mask R that provides a secure channel.

F Oblivious Primitives Obtained by the Framework

The framework presented in Section 4 page 16 provides a generic way to achieve asymmetric protocols around automated trust negotiation. In this section, we show that the classical oblivious primitives directly lie in this framework and can be seen as examples of Oblivious Language-Based Envelope. We in particular show how by tweaking the languages (and choosing appropriate smooth projective hash functions), one can achieve current instantiations of Oblivious Signature-Based Envelope or Oblivious Transfer. We also show how to give a new instantiation of Access Controlled Oblivious Transfer under classical assumptions.

For the specific instantiations, without loss of generality, we are going to focus on elliptic cryptography, but as shown in the case of Oblivious Transfer in [BC15], most building blocks behave the same way under other families of assumptions. As we focus on elliptic curve instantiations, we are going to use ElGamal encryption as the CPA encryption used for the pre-flow.

F.1 Oblivious Signature-Based Envelope

Ideal Functionality for Oblivious Signature-Based Envelope. In order to obtain OSBE as a special case of OLBE, we assume $n = n_{\max} = 1$ and we consider the inner language $\mathfrak{L}$ of valid signatures of a public message M under a given public verification key vk. The corresponding private signing key sk is the language secret key $\mathsf{sk}_{\mathfrak{L}}$, and we assume the existence of a trapdoor key $\mathsf{tk}_{\mathfrak{L}}$ which allows to sample new signatures without the knowledge of $\mathsf{sk}_{\mathfrak{L}}$[15]. Plugging these values in the previous framework directly gives the ideal functionality $\mathcal{F}_{\mathrm{OSBE}}$ in the Simple UC framework, presented in Figure 11, which has never been explicitly given before, to the best of our knowledge.

The functionality $\mathcal{F}_{\mathrm{OSBE}}$ is parametrized by a security parameter $\mathfrak{K}$ and a signature scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1, \ldots, \mathfrak{P}_N$ via the following queries:

- **Upon receiving an input $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m)$ from party $\mathfrak{P}_i$**, with $m \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m)$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further $\mathtt{Send}$-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, \sigma)$ from party $\mathfrak{P}_j$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m)$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$ and send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m')$ to $\mathfrak{P}_j$ where $m' = m$ if $\mathsf{Verify}(\mathsf{vk}, \sigma, M)$ returns 1, and $m' = \perp$ otherwise. Ignore further $\mathtt{Receive}$-message with the same ssid from $\mathfrak{P}_j$.

Fig. 11. Ideal Functionality for Oblivious Signature-Based Envelope $\boldsymbol{\mathcal{F}}_{\mathrm{OSBE}}$

	Generic OSBE	Instantiation Proposal
	Signature and inner language	Waters Signature
	sk, vk	h^x, g^x
$\sigma(m) =$	$\mathsf{Sign}(\mathsf{sk}, m)$	$h^x(u_0 \prod u_i^{m_i})^s, g^s$
$\mathfrak{L}_m =$	$\{\sigma \mid \mathsf{Verify}(\mathsf{vk}, \sigma, m) = 1\}$	$\{\boldsymbol{\sigma} \mid e(\sigma_1, g^x) \cdot e(\sigma_2, F(m)) = e(h, g^x)\}$
$\mathsf{sk}_{\mathfrak{L}} =$	sk	h^x
$\mathsf{tk}_{\mathfrak{L}} =$	allows to compute a new signature	$\log_g h$
	Compatible CCA Encryption	Linear Cramer Shoup Encryption
$\mathsf{ek} =$	ek	$(h, u, v, w, c_1, d_1, c_2, d_2)$
$\mathcal{C} =$	$\mathsf{Encrypt}(\mathsf{ek}, \sigma; \rho)$	$h^{r+t} \cdot h^x F(m)^s, u^r, v^t, w^{r+t},$
		$(c_1 d_1^{\theta})^r (c_2 d_2^{\theta})^t, g^s$
		where $\theta = \mathcal{H}(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4)$
	SPHF	SPHF
$\mathsf{hk} =$	$\mathsf{HashKG}(\mathsf{ek}, \mathfrak{L}_m)$	$\alpha, \beta, \lambda, \mu, \nu \xleftarrow{\$} \mathbb{Z}_p$
$\mathsf{hp} =$	$\mathsf{ProjKG}(\mathsf{hk}, \mathsf{ek}, \mathfrak{L}_m, \mathcal{C})$	$h^\alpha u^\beta w^\mu (c_1 d_1^{\theta})^\nu, h^\alpha v^\lambda w^\mu (c_2 d_2^{\theta})^\nu$
$H =$	$\mathsf{Hash}(\mathsf{hk}, \mathsf{ek}, \mathfrak{L}_m, \mathcal{C})$	$e(\mathcal{C}_2, g)^\beta e(\mathcal{C}_3, g)^\lambda e(\mathcal{C}_4, g)^\mu e(\mathcal{C}_5, g)^\nu$
		$(e(\mathcal{C}_1, g)/(e(h, g^x) e(F(M), \mathcal{C}_6)))^\alpha$
$H' =$	$\mathsf{ProjHash}(\mathsf{hp}, \mathsf{ek}, \mathfrak{L}_m, \rho)$	$e(\mathsf{hp}_1^r \cdot \mathsf{hp}_2^t, g)$

Fig. 12. Setting the Language to instantiate an OSBE, SPHF is based on [BPV12]

Generic Construction and Pairing-Based Instantiation. We give in the left part of Figure 12 the building blocks for OSBE, which directly give a generic construction for OSBE by plugging them into our generic construction for OLBE in Section 4.4.

Instantiating them with Waters' signature scheme [Wat05] and Linear Cramer Shoup encryption [CKP07,Sha07], it gives us a first pairing-based instantiation for OSBE, depicted in the right part of Figure 12, secure in the UC framework against adaptive corruptions (assuming reliable erasures).

Interestingly, this scheme ends up being very similar to the one presented in [BPV12] and proven secure in the standard (not UC) security model, which shows that their scheme remains indeed secure in the UC framework. What was lacking in their paper was the existence (and knowledge) of the trapdoor $\mathsf{tk}_{\mathfrak{L}}$ which, as we can see in the generic proof for OLBE (Appendix E), allows the simulator to be able to always send a valid signature in the first flow on behalf of the receiver. This enables it to be prepared to face adaptive corruptions, whatever input the environment gives it later on.

F.2 1-out-of-n Oblivious Transfer

Ideal Functionality for 1-out-of-n Oblivious Transfer. The ideal functionality of a 1-out-of-n Oblivious Transfer (OT) protocol, from [Can01,CKWZ13,ABB$^+$13], is depicted in Figure 13

[15] Note that $\mathsf{tk}_{\mathfrak{L}}$ may not directly lead to sk but possibly to an alternative signing algorithm.

(adapted to the Simple UC framework). As explained in Section 3.1 page 9, one can easily see it as a special application of our generic OLBE ideal functionality, considering $n \geq 2$ and $n_{\max} = 1$. Indeed, the only change we have to make is not to consider anymore the line numbers as simple numbers, but to "encode" them (the exact encoding will depend on the protocol). For every line i, the language $\mathfrak{L}_i$ will correspond to a representation of line i. Instead of directly giving s to the functionality, the receiver will give it a word $W_s \in \mathfrak{L}_s$. This leads to the functionality given in Figure 2 page 10.

The functionality $\mathcal{F}_{(1,n)\text{-OT}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1,\ldots,\mathfrak{P}_N$ via the following queries:

– **Upon receiving an input $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ from $\mathfrak{P}_i$**, with $m_k \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathscr{S}$. Ignore further $\mathtt{Send}$-message with the same ssid from $\mathfrak{P}_i$.

– **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, s)$ from $\mathfrak{P}_j$**, with $s \in \{1, \ldots, n\}$: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \ldots, m_n))$ is not recorded; otherwise reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to $\mathscr{S}$, send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, m_s)$ to $\mathfrak{P}_j$ and ignore further $\mathtt{Receive}$-message with the same ssid from $\mathfrak{P}_j$.

Fig. 13. Ideal Functionality for 1-out-of-n Oblivious Transfer $\boldsymbol{\mathcal{F}_{(1,n)\text{-OT}}}$

	Generic 1-out-of-n Oblivious Transfer	Possible Instantiation [ABB+13]
	Inner language	Chameleon Hashed Numbers
	Verifiable Chameleon Hash Keys $(\mathsf{ck}, \mathsf{tk}, \mathsf{vtk})$	$(g_1, g \overset{\$}{\leftarrow} \mathbb{G}_2), \alpha = \log_g g_1$,
$c =$	$\mathsf{Commit}_{\mathsf{ck},\mathcal{V}}(s)$	$\forall i$ such that $s = \|\|_{i=1}^{\log n} s_i$:
		$a_i = g_1^{r_{i,s_i}} g^{s_i}, \boldsymbol{r}_i = (r_{i,0}, r_{i,1})$
$\mathfrak{L}_s =$	$\{c \| \mathsf{Verify}(\mathsf{vtk}, c, s) = 1\}$	$\{(\boldsymbol{a}, \boldsymbol{r}) \| \forall i, a_i = g_1^{r_{i,s_i}} g^{s_i}\}$
$\mathsf{sk}_{\mathfrak{L}} =$	$\perp$	$\perp$
$\mathsf{tk}_{\mathfrak{L}} =$	tk	$\log_g h$
	Compatible CCA Encryption	Cramer Shoup Encryption
$\mathsf{ek} =$	ek	$h, u, v, c, d, f \overset{\$}{\leftarrow} \mathbb{G}_1$
$\mathcal{C} =$	$\mathsf{Encrypt}(\mathsf{ek}, c; \rho)$	$\forall i, b, (h^{\rho_{i,b}} \cdot h^x f^{r_{i,b}}, u^{\rho_{i,b}}, v^{s_{\rho,b}},$
		$(cd^\theta)^{s_{\rho,b}}, a_i)$
		where $\theta = \mathcal{H}(\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3)$
	SPHF	SPHF
$\mathsf{hk}_s =$	$\mathsf{HashKG}(\mathsf{ek}, \mathfrak{L}_s)$	$\alpha_s, \beta_s, \mu_s, \nu_s \overset{\$}{\leftarrow} \mathbb{Z}_p$
$\mathsf{hp}_s =$	$\mathsf{ProjKG}(\mathsf{hk}_s, \mathsf{ek}, \mathfrak{L}_s, \mathcal{C})$	$h^{\alpha_s} u^{\beta_s} v^{\mu_s} (cd^\theta)^{\nu_s}, \epsilon_s \overset{\$}{\leftarrow} \mathbb{Z}_p$
$H =$	$\mathsf{Hash}(\mathsf{hk}_s, \mathsf{vtk}, \mathsf{ek}, \mathfrak{L}_s, \mathcal{C})$	$\prod_i ((e(\mathcal{C}_{i,s_i,1}/g^{s_i}, g_1)/e(f, a_i))^{\alpha_s}$
		$\mathcal{C}_{i,s_i,2}^{\beta_s} \mathcal{C}_{i,s_i,3}^{\mu_s} \mathcal{C}_{i,s_i,4}^{\nu_s} \Big)^{\epsilon_s^i}$
$H' =$	$\mathsf{ProjHash}(\mathsf{hp}_s, \mathsf{ek}, \mathfrak{L}_s, \rho)$	$e(\prod_i \mathsf{hp}^{\rho_{i,s_i} \epsilon_s^i}, g_1)$

Fig. 14. Instantiating a 1-out-n Oblivious Transfer, SPHF is derived using the framework from [BBC+13a]

Generic Construction and Pairing-Based Instantiation. From this small change, we can give a generic construction which directly falls from our generic construction for OLBE. We consider, for every line i, the inner language $\mathfrak{L}_i$ of valid chameleon hashes of the line $i \in \{1, \ldots, k\}$ under the corresponding chameleon hash key ck and the verification key vk, given by the authority (see Section A page 25 for the definitions for chameleon hash). In this case, the language is not keyed (anyone can sample a word in it), but it possesses a trapdoor $\mathsf{tk}_{\mathfrak{L}} = \mathsf{tk}$ which is the trapdoor of the chameleon hash. This trapdoor enables the simulator to send a chameleon hash which can correspond to any line s which is exactly what is required in our proof of generic OLBE (see Appendix E).

Indeed, as explained in Section 3, when dealing with 1-out-of-n Oblivious Transfer, one usually considers a database with various lines, so the naive approach to fit in our framework would have been to consider a set of languages being the set of numerals $\{\{1\}, \dots, \{n\}\}$, expecting the receiver to commit to a word in one of these languages $\mathfrak{L}_i$, so that he recovers the corresponding line number i of the database. But a drawback with this approach is that in case of an adaptive corruption of the receiver, one would need to be able to equivocate the commitment to the word, which would require for instance a commitment at least extractable and equivocable (and even SPHF-friendly, see [ABB+13]), and not a simple $\mathsf{CCA}-2$ encryption as we propose in our generic construction of OLBE. This is the reason why we propose to consider the set of languages composed of valid representations of the corresponding integer, which allows more freedom in the constructions. Each line is now indexed by a whole language instead of a single number.

F.3 k-out-of-n Oblivious Transfer

Ideal Functionality for k-out-of-n Oblivious Transfer. From the generic framework of OLBE and its adaptation to 1-out-of-n Oblivious Transfer described right above in the previous section, one easily gets the ideal functionality of a k-out-of-n Oblivious Transfer (OT) protocol, by simply letting $n_{\max} = k \in [\![1; n]\!]$. This leads to the ideal functionality depicted in Figure 15 (rewritten in Simple UC).

The functionality $\mathcal{F}_{(k,n)\text{-OT}}$ is parametrized by a security parameter $\mathfrak{K}$ and a set of languages $(\mathfrak{L}_1, \dots, \mathfrak{L}_n)$ along with the corresponding public verification algorithms $(\mathsf{Verify}_1, \dots, \mathsf{Verify}_n)$. It interacts with an adversary $\mathscr{S}$ and a set of parties $\mathfrak{P}_1$,...,$\mathfrak{P}_N$ via the following queries:

- **Upon receiving an input $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \dots, m_n))$ from party $\mathfrak{P}_i$**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \dots, m_n))$ and reveal $(\mathtt{Send}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$. Ignore further $\mathtt{Send}$-message with the same ssid from $\mathfrak{P}_i$.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (W_i)_{i\in I})$ where $I \subset \{1, \dots, n\}$ and $|I| = k$ from party $\mathfrak{P}_j$:** ignore the message if $(\mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m_1, \dots, m_n))$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j)$ to the adversary $\mathscr{S}$, send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, \mathfrak{P}_i, \mathfrak{P}_j, (m'_i)_{i\in I})$ to $\mathfrak{P}_j$ where $m'_i = m_i$ if $\mathsf{Verify}_i(W_i, \mathfrak{L}_i)$ returns 1, and $m'_i = \perp$ otherwise. Ignore further $\mathtt{Receive}$-message with the same ssid from $\mathfrak{P}_j$.

Fig. 15. Ideal Functionality for k-out-of-n Oblivious Transfer $\mathcal{F}_{(k,n)\text{-OT}}$

Generic Construction and Pairing-Based Instantiation. One can obtain a generic construction by simply applying the 1-out-of-n OT generic construction described above in parallel k times together with a NIZK made by the sender proving that all sets of messages are the same. This also leads to a pairing-based instantiation (while using Groth Sahai proof as NIZK for example [GS08]).

F.4 Conditioned Oblivious Transfer

Ideal Functionality for Conditioned Oblivious Transfer. Oblivious Transfer protocols allow the user to query a line in a database without restriction: The user automatically gets the line he asks for. While this has already found many applications, one may consider a slightly more complicated problem in which each line of the database has some additional access restriction (such as credentials, accreditation level, ...). For instance, lines numbers 2 to 10 of a database may be only accessible to the members of an organisation, while line number 1 may be only accessible to the presidents of several associations.

This problem has already been studied in various forms: Priced Oblivious Transfer (introduced in [AIR01]), which allows the user to access a given line if he has enough cash to pay (so if his balance is greater than the line price); Conditional Oblivious Transfer (formally introduced in [DOR99] but with few conditions about user privacy); Access Control Oblivious Transfer [CDN09], which aims at solving this issue for users with credentials.

Our framework allows to supersede all these primitives[16]: Thanks to our generalization of the Oblivious Transfer ideal functionality (using languages instead of simple numbers for the lines requested), the ideal functionality remains the same, since the languages already allow to add the credentials, signatures, prices, etc., required by those specific forms of OT.

Generic Constructions and Pairing-Based Instantiations. Our framework also provides a means to obtain tightly secure instantiations under classical (non-interactive, non q-type) assumptions. The required line restrictions need to be compatible with each other (if the word required for a line needs 2 elements to be committed, while another line requires words that can only be committed with 20 elements, there is no way to equivocate the first one to the second one, except if we adapt the language for the first line to have *dead* commitment terms so that all the lines require the same commitment size).

	Generic 1-out-of-n Conditioned Oblivious Transfer
	Inner language
	Trapped Verifiable Commitment Keys $(\mathsf{ck}, \mathsf{tk})$
$c =$	$\mathsf{Commit}_{\mathcal{V}}(s)$
$\mathfrak{L}_s =$	$\{c \mid \mathsf{Verify}(c, s) = 1 \wedge c \in \{\text{Access Requirement s}\}\}$
$\mathsf{sk}_{\mathfrak{L}} =$	$\{\mathsf{sk}_s\}$
$\mathsf{tk}_{\mathfrak{L}} =$	$\mathsf{tk}_{\mathfrak{L}}$
	Compatible CCA Encryption
$\mathsf{ek} =$	ek
$\mathcal{C} =$	$\mathsf{Encrypt}(\mathsf{ek}, c; \rho)$
	SPHF
$\mathsf{hk}_s =$	$\mathsf{HashKG}(\mathsf{ek}, \mathfrak{L}_s)$
$\mathsf{hp}_s =$	$\mathsf{ProjKG}(\mathsf{hk}_s, \mathsf{ek}, \mathfrak{L}_s, \mathcal{C})$
$H =$	$\mathsf{Hash}(\mathsf{hk}_s, \mathsf{ek}, \mathfrak{L}_s, \mathcal{C})$
$H' =$	$\mathsf{ProjHash}(\mathsf{hp}_s, \mathsf{ek}, \mathfrak{L}_s, \rho)$

Fig. 16. Instantiating a 1-out-n Conditioned Oblivious Transfer

This could encompass a lot of different controls: For example, each line can be protected by a different password; a line can require a signature (credential) from a given authority on the line number, on the user identity, on a pseudonym; one can imagine more complicated policies like a required hamming weight, a scalar product, range, . . . or any combination of those.

Giving a specific instantiation for all those cases is out of the scope of this article but for example, priced Oblivious Transfer requires to prove that the balance is greater than the line price[17]. The easiest way to do so would require to use an SPHF proving that the balance does not belong to the range $[\![0, \text{price}]\!]$. Both range proofs (adapting [BG13] with SPHF polynomial evaluations) and proof of negativity of a statement are already part of the SPHF toolbox.

[16] Although some of them (like Priced Oblivious Transfer) may require additional machinery to (privately) keep track of each user's balance over the course of many transactions.

[17] As already stated in Note 16, it also requires some additional machinery to (privately) keep track of each user's balance over the course of many transactions.

Appendix E

Structure-Preserving Smooth Projective Hashing [BC16]

This is the Full Version of the Extended Abstract that appears in Advances in Cryptology — Proceedings of ASIACRYPT'2016 (4 – 8 December 2016, Hanoi, Vietnam), Jung Hee Cheon and Tsuyoshi Takagi, Springer-Verlag, Part II, LNCS 10032, pages 339–369.

Authors

Olivier Blazy, Céline Chevalier

Abstract

Smooth projective hashing has proven to be an extremely useful primitive, in particular when used in conjunction with commitments to provide implicit decommitment. This has lead to applications proven secure in the UC framework, even in presence of an adversary which can do adaptive corruptions, like for example Password Authenticated Key Exchange (PAKE), and 1-out-of-m Oblivious Transfer (OT). However such solutions still lack in efficiency, since they heavily scale on the underlying message length.

Structure-preserving cryptography aims at providing elegant and efficient schemes based on classical assumptions and standard group operations on group elements. Recent trend focuses on constructions of structure-preserving signatures, which require message, signature and verification keys to lie in the base group, while the verification equations only consist of pairing-product equations. Classical constructions of Smooth Projective Hash Function suffer from the same limitation as classical signatures: at least one part of the computation (messages for signature, witnesses for SPHF) is a scalar.

In this work, we introduce and instantiate the concept of Structure-Preserving Smooth Projective Hash Function, and give as applications more efficient instantiations for one-round PAKE and three-round OT, and information retrieval thanks to Anonymous Credentials, all UC-secure against adaptive adversaries.

1 Introduction

Smooth Projective Hash Functions (SPHF) were introduced by Cramer and Shoup [CS02] as a means to design chosen-ciphertext-secure public-key encryption schemes. These hash functions are defined such as their value can be computed in two different ways if the input belongs to a particular subset (the *language*), either using a private hashing key or a public projection key along with a private witness ensuring that the input belongs to the language.

In addition to providing a more intuitive abstraction for their original public-key encryption scheme in [CS98], the notion of SPHF also enables new efficient instantiations of their scheme under different complexity assumptions such as DLin, or more generally $k-\mathsf{MDDH}$. Due to its usefulness, the notion of SPHF was later extended to several interactive contexts. One of the most classical applications is to combine them with commitments in order to provide *implicit* decommitments.

Commitment schemes have become a central tool used in cryptographic protocols. These two-party primitives (between a committer and a receiver) are divided into two phases. First, in the *commit* phase, the committer gives the receiver an analogue of a sealed envelope containing a value m, while later in the *opening* phase, the committer reveals m in such a way that the receiver can verify whether it was indeed m that was contained in the envelope. In many applications, for example password-based authenticated key-exchange, in which the committed value is a password, one wants the opening to be implicit, which means that the committer does not really open its commitment, but rather convinces the receiver that it actually committed to the value it pretended to.

An additional difficulty arises when one wants to prove the protocols in the universal composability framework proposed in [Can01]. Skipping the details, when the protocol uses commitments, this usually forces those commitments to be simultaneously *extractable* (meaning that a simulator can recover the committed value m thanks to a trapdoor) and *equivocable* (meaning that a simulator can open a commitment to a value m' different from the committed value m thanks to a trapdoor), which is quite a difficult goal to achieve.

Using SPHF with commitments to achieve an implicit decommitment, the language is usually defined on group elements, with projection keys being group elements, and witnesses being scalars. While in several applications, this has already lead to efficient constructions, the fact that witnesses have to be scalars (and in particular in case of commitments, the randomness used to commit) leads to drastic restrictions when trying to build protocols secure against adaptive corruptions in the UC framework.

This is the classical paradigm of protocol design, where generic primitives used in a modular approach lead to a simple design but quite inefficient constructions, while when trying to move to ad-hoc constructions, the conceptual simplicity is lost and even though efficiency might be gained, a proper security proof gets trickier. Following the same kind of reasoning, [AFG+10] introduced the concept of structure-preserving signatures in order to take the best of both worlds. There has been an ongoing series of work surrounding this notion, for instance [AGHO11, ACD+12, ADK+13, AGOT14a, AGOT14b]. This has shown that structure-preserving cryptography indeed provides the tools needed to have simultaneously simple and efficient protocols.

1.1 Related Work

Smooth Projective Hash Functions (SPHF) were introduced by Cramer and Shoup [CS02] and have been widely used since then, for instance for password-authenticated key exchange (PAKE) [GL03, ACP09, KV09, KV11, BBC+13b], or oblivious transfer (OT) [Kal05, CKWZ13, ABB+13], and a classification was introduced separating SPHF into three main kinds, KV-SPHF, CS-SPHF, GL-SPHF depending on how the projection keys are generated and when, the former allowing one-round protocols, while the latter have more efficient communication costs (see Section 2.2).

Password-Authenticated Key Exchange (PAKE) protocols were proposed in 1992 by Bellovin and Merritt [BM92] where authentication is done using a simple password, possibly drawn from a small entropy space subject to exhaustive search. Since then, many schemes have been proposed and studied. SPHF have been extensively used, starting with the work of Gennaro and Lindell [GL03] which generalized an earlier construction by Katz, Ostrovsky, and Yung [KOY01], and followed by several other works [CHK$^+$05, ACP09]. More recently, a variant of SPHF proposed by Katz and Vaikuntanathan even allowed the construction of one-round PAKE schemes [KV11, BBC$^+$13b]. The most efficient PAKE scheme so far (using completely different techniques) is the recent Asiacrypt paper [JR14].

The first ideal functionality for PAKE protocols in the UC framework [Can01, CK02] was proposed by Canetti *et al.* [CHK$^+$05], who showed how a simple variant of the Gennaro-Lindell methodology [GL03] could lead to a secure protocol. Though quite efficient, their protocol was not known to be secure against adaptive adversaries, that are capable of corrupting players at any time, and learn their internal states. The first ones to propose an adaptively secure PAKE in the UC framework were Barak *et al.* [BCL$^+$05] using general techniques from multi-party computation. Though conceptually simple, their solution results in quite inefficient schemes.

Recent adaptively secure PAKE were proposed by Abdalla *et al.* [ACP09, ABB$^+$13], following the Gennaro-Lindell methodology with variation of the Canetti-Fischlin commitment [CF01]. However their communication size is growing in the size of the passwords, which is leaking information about an upper-bound on the password used in each exchange.

Oblivious Transfer (OT) was introduced in 1981 by Rabin [Rab81] as a way to allow a receiver to get exactly one out of k messages sent by another party, the sender. In these schemes, the receiver should be oblivious to the other values, and the sender should be oblivious to which value was received. Since then, several instantiations and optimizations of such protocols have appeared in the literature, including proposals in the UC framework [NP01, CLOS02].

More recently, new instantiations have been proposed, trying to reach round-optimality [HK07], or low communication costs [PVW08]. The 1-out-of-2 OT scheme by Choi *et al.* [CKWZ13] based on the DDH assumption seems to be the most efficient one among those that are secure against adaptive corruptions in the CRS model with erasures. But it does not scale to 1-out-of-m OT, for $m > 2$. [ABB$^+$13, BC15] proposed a generic construction of 1-out-of-m OT secure against adaptive corruptions in the CRS model, however the commitment was still growing in the logarithm of the database length. While this is not so much a security issue for OT as this length is supposed to be fixed at the start of the protocol, this is however a weak spot for the efficiency of the final construction.

1.2 Our contributions

Similarly to structure-preserving signatures requiring the message, the signature, and the public keys to be group elements, we propose in this paper the notion of structure-preserving Smooth Projective Hash Functions (SP-SPHF), where both words, witnesses and projection keys are group elements, and hash and projective hash computations are doable with simple pairing-product equations in the context of bilinear groups.

This allows, for example, to build Smooth Projective Hash Functions that implicitly demonstrate the knowledge of a Groth Sahai Proof (serving as a witness).

We show how to transform every previously known pairing-less construction of SPHF to fit this methodology, and then propose several applications in which storing a group element as a witness allows to avoid the drastic restrictions that arise when building protocols secure against adaptive corruptions in the UC framework with a scalar as witness. Asking the witness to be a group element enables us to gain more freedom in the simulation (the discrete logarithm of this element and / or real extraction from a commitment). For instance, the simulator can always commit honestly to a random message, since it only needs to modify its witness in the equivocation phase. Furthermore, it allows to avoid bit-per-bit construction. Such design carries similarity with the publicly verifiable MACs from [KPW15], where the pairing operation allows to relax the verification procedure.

A work from Jutla and Roy has appeared on eprint [JR16] considering a parallel between QA-NIZK and SPHF: Independently from ours, they define a transformation from one to another. Their transformation can then be extended to view QA-NIZK as a special case of SP-SPHF, and so be encompassed by our framework.

As an example, we show that the UC-commitment from [FLM11] (while not fitting with the methodology of traditional SPHF from [ABB+13]), is compatible with SP-SPHF and can be used to build UC protocols. As a side contribution, we first generalize this commitment from DLin to the $k-\mathsf{MDDH}$ assumption from [EHK+13]. The combination of this commitment and the associated SP-SPHF then enables us to give three interesting applications.

Adaptively secure 1-out-of-m Oblivious Transfer. First, we provide a construction of a three-round UC-secure 1-out-of-m OT. Assuming reliable erasures and a single global CRS, we show in Section 5 that our instantiation is UC-secure against adaptive adversaries. Besides having a lesser number of rounds than most recent existing OT schemes with similar security levels, our resulting protocol also has a better communication complexity than the best known solutions so far [CKWZ13, ABB+13] (see Table 1 for a comparison). For ease of readability, we emphasize in this table the SXDH communication cost[1], which is simply k-MDDH for $k = 1$. Our protocol is "nearly optimal" in the sense that it is still linear in the number of lines m, but the constant in front of m is 1.

Table 1. Comparison with existing UC-secure OT schemes

	Flow	Communication Complexity	Assumption	1-out-of
[CKWZ13]	4	$26\ \mathbb{G} + 7\ \mathbb{Z}_p$	DDH	2
[ABB+13]	3	$(m + 8\log m) \times \mathbb{G}_1 + \log m \times \mathbb{G}_2 + 1 \times \mathbb{Z}_p$	SXDH	m
This paper	3	$(k+3) \times \mathbb{G}_1 + (2 + (3+k)m + k(k+1)) \times \mathbb{G}_2 + m \times \mathbb{Z}_p$	$k-\mathsf{MDDH}$	m
This paper	3	$4 \times \mathbb{G}_1 + 12 \times \mathbb{G}_2 + 2 \times \mathbb{Z}_p$	SXDH	2

One-round adaptively secure PAKE. Then, we provide an instantiation of a one-round UC-secure PAKE under any $k-\mathsf{MDDH}$ assumption. Once again, we show in Section 6 that the UC-security holds against adaptive adversaries, assuming reliable erasures and a single global CRS. Contrarily to most existing one-round adaptively secure PAKE, we show that our scheme enjoys a much better communication complexity while not leaking information about the length of the password used (see Table 2 for a comparison, in particular for the SXDH version). Only [JR14] achieves a slightly better complexity as ours, but only for SXDH, while ours easily extends to $k-\mathsf{MDDH}$. Furthermore, our construction is an extension to SP-SPHF of well-known classical constructions based on SPHF, which makes it simpler to understand. We omit [BC15] from the following table, as its contribution is to widen the construction to non-pairing based hypotheses.

Anonymous Credential-Based Message Transmission. Typical credential use involves three main parties. Users need to interact with some authorities to obtain their credentials (assumed to be a set of attributes validated / signed), and then prove to a server that a subpart of their attributes verifies an expect policy. We present a constant-size, round-optimal protocol that allow to use a Credential to retrieve a message without revealing the Anonymous Credentials in a UC secure way, by simply building on the technique proposed earlier in the paper.

[1] Our OT and PAKE protocols are described in k-MDDH but one directly obtains the SXDH versions by simply letting $k = 1$ in the commitment presented in Section 4.2 (see Appendix **??** for details).

Table 2. Comparison with existing UC-secure PAKE schemes where $|\text{password}| = m$

	Adaptive	One-round	Communication complexity	Assumption
[ACP09]	yes	no	$2 \times (2m + 22m\mathfrak{K}) \times \mathbb{G} + \text{OTS}$	DDH
[KV11]	no	yes	$\approx 2 \times 70 \times \mathbb{G}$	DLIN
[BBC+13b]	no	yes	$2 \times 6 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH
[ABB+13]	yes	yes	$2 \times 10m \times \mathbb{G}_1 + 2 \times m \times \mathbb{G}_2$	SXDH
[JR14]	yes	yes	$4 \times \mathbb{G}_1 + 4 \times \mathbb{G}_2$	SXDH
this paper	yes	yes	$2 \times (k+3) \times \mathbb{G}_1$ $+2 \times (k+3+k(k+1)) \times \mathbb{G}_2$	k-MDDH
this paper	yes	yes	$2 \times 4 \times \mathbb{G}_1 + 2 \times 5 \times \mathbb{G}_2$	SXDH

2 Definitions

2.1 Notations

If $\boldsymbol{x} \in \mathcal{S}^n$, then $|\boldsymbol{x}|$ denotes the length n of the vector, and by default vectors are assumed to be column vectors. Further, $x \overset{\$}{\leftarrow} \mathcal{S}$ denotes the process of sampling an element x from the set $\mathcal{S}$ uniformly at random.

2.2 Primitives

Encryption. An encryption scheme $\mathcal{C}$ is described by four algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$), defined formally in Appendix A.1.

Commitments. We refer the reader to [ABB+13] for formal definitions and results but we give here an informal overview to help the unfamiliar reader with the following. A *non-interactive labelled commitment scheme* $\mathcal{C}$ is defined by three algorithms:

- $\mathsf{SetupCom}(1^{\mathfrak{K}})$ takes as input the security parameter $\mathfrak{K}$ and outputs the global parameters, passed through the CRS ρ to all other algorithms;
- $\mathsf{Com}^{\ell}(x)$ takes as input a label ℓ and a message x, and outputs a pair (C, δ), where C is the commitment of x for the label ℓ, and δ is the corresponding opening data (a.k.a. decommitment information). This is a probabilistic algorithm.
- $\mathsf{VerCom}^{\ell}(C, x, \delta)$ takes as input a commitment C, a label ℓ, a message x, and the opening data δ and outputs 1 (true) if δ is a valid opening data for C, x and ℓ. It always outputs 0 (false) on $x = \bot$.

The basic properties required for commitments are *correctness* (for all correctly generated CRS ρ, all commitments and opening data honestly generated pass the verification VerCom test), the *hiding property* (the commitment does not leak any information about the committed value) and the *binding property* (no adversary can open a commitment in two different ways). More complex properties (equivocability and extractability) are required by the UC framework and described in Appendix A.2 for lack of space.

Smooth Projective Hash Functions. SPHF were introduced by Cramer and Shoup [CS02] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. The notion of SPHF has already found numerous applications in various contexts in cryptography (*e.g.* [GL03, Kal05, ACP09, BPV12]).

Definition 1 (Smooth Projective Hashing System). *A Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$, is defined by five algorithms* ($\mathsf{Setup}, \mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash}$)*:*

– $\mathsf{Setup}(1^{\mathfrak{K}})$ *generates the global parameters* param *of the scheme, and the description of an* $\mathcal{NP}$ *language* $\mathfrak{L}$
– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$*, outputs a hashing key* hk *for the language* $\mathfrak{L}$*;*
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$*, derives the projection key* hp*, using the hashing key* hk*,*
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$*, outputs a hash value v, thanks to the hashing key* hk*, and W,*
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$*, outputs the hash value v', thanks to* hp *and the witness w that $W \in \mathfrak{L}$.*

In the following, we consider $\mathfrak{L}$ as a hard-partitioned subset of X, *i.e.* it is computationally hard to distinguish a random element in $\mathfrak{L}$ from a random element in $X \setminus \mathfrak{L}$.

A Smooth Projective Hash Function SPHF should satisfy the following properties:

– *Correctness*: Let $W \in \mathfrak{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have

$$\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) = \mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w).$$

– *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \in \mathbb{G} \end{array} \right\}$$
$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathbb{G} \end{array} \right\}.$$

A third property called *Pseudo-Randomness*, is implied by the Smoothness on Hard Subset membership languages. If $W \in \mathfrak{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time the following advantage is negligible

$$\mathsf{Adv}^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(\mathfrak{K}) = |\Pr_{\Delta_1}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) = 1]|$$

In [BBC$^+$13b], the authors introduced a new notation for SPHF: for a language $\mathfrak{L}$, there exist a function Γ and a family of functions Θ, such that $\boldsymbol{u} \in \mathfrak{L}$, if and only if, $\Theta(\boldsymbol{u})$ is a linear combination $\boldsymbol{\lambda}$ of the rows of $\Gamma(\boldsymbol{u})$. We furthermore require that a user, who knows a witness of the membership $\boldsymbol{u} \in \mathfrak{L}$, can efficiently compute the linear combination $\boldsymbol{\lambda}$. The SPHF can now then be described as:

– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, outputs a hashing key $\mathsf{hk} = \boldsymbol{\alpha}$ for the language $\mathfrak{L}$,
– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \boldsymbol{u})$, derives the projection key $\mathsf{hp} = \boldsymbol{\gamma}(\boldsymbol{u})$,
– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), \boldsymbol{u})$, outputs a hash value $H = \Theta(\boldsymbol{u}) \odot \boldsymbol{\alpha}$,
– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), \boldsymbol{u}, \boldsymbol{\lambda})$, outputs the hash value $H' = \boldsymbol{\lambda} \odot \boldsymbol{\gamma}(\boldsymbol{u})$.

In the special case where $\mathsf{hp} = \boldsymbol{\gamma}(\boldsymbol{u}) = \boldsymbol{\gamma}$, we speak about KV-SPHF when the projection key can be given before seeing the word $\boldsymbol{u}$, and of CS-SPHF, when the projection key while independent of the word is given after seeing it. (In reference to [KV11, CS02] where those kinds of SPHF were first use). We give in Section 3.3 an example of KV-SPHF for Cramer-Shoup encryption, both in classical and new notations.

We will need a third property for our one-round PAKE protocol. This property, called strong pseudo-randomness in [BBC$^+$13b], is recalled in Appendix A.3 for lack of space.

2.3 Building Blocks

Decisional Diffie-Hellman (DDH) The Decisional Diffie-Hellman hypothesis says that in a multiplicative group $(p, \mathbb{G}, g)$ when we are given $(g^\lambda, g^\mu, g^\psi)$ for unknown random $\lambda, \mu, \psi \overset{\$}{\leftarrow} \mathbb{Z}_p$, it is hard to decide whether $\psi = \lambda \times \mu$.

Pairing groups. Let GGen be a probabilistic polynomial time (PPT) algorithm that on input $1^{\mathfrak{K}}$ returns a description $\mathcal{G} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of asymmetric pairing groups where $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ are cyclic groups of order p for a $\mathfrak{K}$-bit prime p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ is an efficiently computable (non-degenerated) bilinear map. Define $g_T := e(g_1, g_2)$, which is a generator in $\mathbb{G}_T$.

Matricial Notations. If $\boldsymbol{A} \in \mathbb{Z}_p^{(k+1)\times n}$ is a matrix, then $\overline{\boldsymbol{A}} \in \mathbb{Z}_p^{k\times n}$ denotes the upper matrix of $\boldsymbol{A}$ and $\underline{\boldsymbol{A}} \in \mathbb{Z}_p^{1\times n}$ denotes the last row of $\boldsymbol{A}$. We use classical notations from [GS08] for operations on vectors (. for the dot product and $\odot$ for the product component-wise). Concatenation of matrices having the same number of lines will be denoted by $\boldsymbol{A}||\boldsymbol{B}$ (where $a||b + c$ should be implicitly parsed as $a||(b + c)$).

We use implicit representation of group elements as introduced in [EHK$^+$13]. For $s \in \{1, 2, T\}$ and $a \in \mathbb{Z}_p$ define $[a]_s = g_s^a \in \mathbb{G}_s$ as the *implicit representation* of a in $\mathbb{G}_s$ (we use $[a] = g^a \in \mathbb{G}$ if we consider a unique group). More generally, for a matrix $\boldsymbol{A} = (a_{ij}) \in \mathbb{Z}_p^{n\times m}$ we define $[\boldsymbol{A}]_s$ as the implicit representation of $\boldsymbol{A}$ in $\mathbb{G}_s$:

$$[\boldsymbol{A}]_s := \begin{pmatrix} g_s^{a_{11}} \ldots g_s^{a_{1m}} \\ \\ g_s^{a_{n1}} \ldots g_s^{a_{nm}} \end{pmatrix} \in \mathbb{G}_s^{n\times m}$$

We will always use this implicit notation of elements in $\mathbb{G}_s$, i.e., we let $[a]_s \in \mathbb{G}_s$ be an element in $\mathbb{G}_s$. Note that from $[a]_s \in \mathbb{G}_s$ it is generally hard to compute the value a (discrete logarithm problem in $\mathbb{G}_s$). Further, from $[b]_T \in \mathbb{G}_T$ it is hard to compute the value $[b]_1 \in \mathbb{G}_1$ and $[b]_2 \in \mathbb{G}_2$ (pairing inversion problem). Obviously, given $[a]_s \in \mathbb{G}_s$ and a scalar $x \in \mathbb{Z}_p$, one can efficiently compute $[ax]_s \in \mathbb{G}_s$. Further, given $[a]_1, [b]_2$ one can efficiently compute $[ab]_T$ using the pairing e. For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_p^k$ define $e([\boldsymbol{a}]_1, [\boldsymbol{b}]_2) := [\boldsymbol{a}^\top \boldsymbol{b}]_T \in \mathbb{G}_T$.

If $a \in \mathbb{Z}_p$, we define the $(k+1)$-vector: $\iota_s(a) := (1_s, \ldots, 1_s, [a]_s)$ (this notion can be implicitly extended to vectors $a \in \mathbb{Z}_p^n$), and the $k+1$ by $k+1$ matrix $\iota_T(a) := \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & 1 \\ 1 & 1 & a \end{pmatrix}$.

Assumptions. We recall the definition of the matrix Diffie-Hellman (MDDH) assumption given in [EHK$^+$13].

Definition 2 (Matrix Distribution). *Let $k \in \mathbb{N}$. We call $\mathcal{D}_k$ a matrix distribution if it outputs matrices in $\mathbb{Z}_p^{(k+1)\times k}$ of full rank k in polynomial time.*

Without loss of generality, we assume the first k rows of $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_k$ form an invertible matrix. The $\mathcal{D}_k$-Matrix Diffie-Hellman problem is to distinguish the two distributions $([\mathbf{A}], [\mathbf{A}\boldsymbol{w}])$ and $([\mathbf{A}], [\boldsymbol{u}])$ where $\mathbf{A} \overset{\$}{\leftarrow} \mathcal{D}_k$, $\boldsymbol{w} \overset{\$}{\leftarrow} \mathbb{Z}_p^k$ and $\boldsymbol{u} \overset{\$}{\leftarrow} \mathbb{Z}_p^{k+1}$.

Definition 3 ($\mathcal{D}_k$-Matrix Diffie-Hellman Assumption $\mathcal{D}_k$-MDDH). *Let $\mathcal{D}_k$ be a matrix distribution and $s \in \{1, 2, T\}$. We say that the $\mathcal{D}_k$-Matrix Diffie-Hellman ($\mathcal{D}_k$-MDDH) Assumption holds relative to GGen in group $\mathbb{G}_s$ if for all PPT adversaries $\mathcal{D}$,*

$$\begin{aligned}\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) &:= |\Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{Aw}]_s) = 1] - \Pr[\mathcal{D}(\mathcal{G}, [\boldsymbol{A}]_s, [\boldsymbol{u}]_s) = 1]| \\ &= \mathsf{negl}(\lambda),\end{aligned}$$

where the probability is taken over $\mathcal{G} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{w} \xleftarrow{\$} \mathbb{Z}_p^k, \boldsymbol{u} \xleftarrow{\$} \mathbb{Z}_p^{k+1}$.

For each $k \geq 1$, [EHK+13] specifies distributions $\mathcal{L}_k$, $\mathcal{U}_k$, ... such that the corresponding $\mathcal{D}_k$-MDDH assumption is the k-Linear assumption, the k-uniform and others. All assumptions are generically secure in bilinear groups and form a hierarchy of increasingly weaker assumptions. The distributions are exemplified for $k = 2$, where $a_1, \ldots, a_6 \xleftarrow{\$} \mathbb{Z}_p$.

$$\mathcal{L}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & 0 \\ 0 & a_2 \\ 1 & 1 \end{pmatrix} \quad \mathcal{U}_2 : \boldsymbol{A} = \begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \\ a_5 & a_6 \end{pmatrix}.$$

It was also shown in [EHK+13] that $\mathcal{U}_k$-MDDH is implied by all other $\mathcal{D}_k$-MDDH assumptions. In the following, we write $k -$ MDDH for $\mathcal{D}_k -$ MDDH.

Lemma 4 (Random self reducibility [EHK+13]). *For any matrix distribution $\mathcal{D}_k$, $\mathcal{D}_k$-MDDH is random self-reducible. In particular, for any $m \geq 1$,*

$$\mathbf{Adv}_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}) + \tfrac{1}{q-1} \geq \mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}')$$

where $\mathbf{Adv}^m_{\mathcal{D}_k,\mathsf{GGen}}(\mathcal{D}') := \Pr[\mathcal{D}'(\mathcal{G}, [\boldsymbol{A}], [\boldsymbol{AW}]) \Rightarrow 1] - \Pr[\mathcal{D}'(\mathcal{G}, [\boldsymbol{A}], [\boldsymbol{U}]) \Rightarrow 1]$, with $\mathcal{G} \leftarrow \mathsf{GGen}(1^\lambda)$, $\boldsymbol{A} \xleftarrow{\$} \mathcal{D}_k, \boldsymbol{W} \xleftarrow{\$} \mathbb{Z}_p^{k\times m}, \boldsymbol{U} \xleftarrow{\$} \mathbb{Z}_p^{(k+1)\times m}$.

Remark: It should be noted that $\mathcal{L}_1, \mathcal{L}_2$ are respectively the SXDH and DLin assumptions that we recall below for completeness.

Definition 5 (Decisional Linear (DLin [BBS04])). *The Decisional Linear hypothesis says that in a multiplicative group $(p, \mathbb{G}, g)$ when we are given $(g^\lambda, g^\mu, g^{\alpha\lambda}, g^{\beta\mu}, g^\psi)$ for unknown random $\alpha, \beta, \lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $\psi = \alpha + \beta$.*

Definition 6 (Symmetric External Diffie Hellman (SXDH [ACHdM05])). *This variant of DDH, used mostly in bilinear groups in which no computationally efficient homomorphism exists from $\mathbb{G}_2$ in $\mathbb{G}_1$ or $\mathbb{G}_1$ to $\mathbb{G}_2$, states that DDH is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

Labelled Cramer-Shoup Encryption. We present here the well-known encryption schemes based on DDH, and we show in Section 4 how to extend it to $\mathcal{D}_k -$ MDDH. We focus on Cramer-Shoup [CS98] in all the following of the paper, but one easily obtains the same results on El Gamal IND-CPA scheme [ElG84] by simply omitting the corresponding parts. We are going to rely on the IND-CCA property to be able to decrypt queries in the simulation.

VANILLA CRAMER-SHOUP ENCRYPTION. The Cramer-Shoup encryption scheme is an IND-CCA version of the ElGamal Encryption. We present it here as a labeled public-key encryption scheme, the classical version is done with $\ell = \emptyset$.

- Setup($1^{\mathfrak{K}}$) generates a group $\mathbb{G}$ of order p, with a generator g
- KeyGen(param) generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a Collision-Resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- Encrypt(ℓ, ek, M; r), for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.

– $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$.
If the equality holds, one computes $M = e/(u_1^z)$ and outputs M. Otherwise, one outputs $\perp$.

The security of the scheme is proven under the DDH assumption and the fact the hash function used is a Universal One-Way Hash Function.

In following work [CS02] they refined the proof, explaining that the scheme can be viewed as a 2-Universal Hash Proof on the language of valid Diffie Hellman tuple.

Vanilla Cramer-Shoup Encryption with Matricial Notations.

– $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with a generator g, with an underlying matrix assumption $\mathcal{D}_1$ using a base matrix $[\boldsymbol{A}] \in \mathbb{G}^{2\times 1}$;
– $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{z} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^2$ (with $\boldsymbol{t}_1 = (x_1, x_2)$, $\boldsymbol{t}_2 = (y_1, y_2)$ and $\boldsymbol{z} = (z, 1)$), and sets $c = \boldsymbol{t}_1\boldsymbol{A}, d = \boldsymbol{t}_2\boldsymbol{A}, h = \boldsymbol{z}\boldsymbol{A}$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = ([A], [c], [d], [h], \mathfrak{H}_K)$.
– $\mathsf{Encrypt}(\ell, \mathsf{ek}, [m]; r)$, for a message $M = [m] \in \mathbb{G}$ and random scalar $r \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = [\boldsymbol{A}r]), e = [\boldsymbol{h}r + m], v = [(\boldsymbol{c} + \boldsymbol{d} \odot \xi)\boldsymbol{r}]$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
– $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether v is consistent with $\boldsymbol{t}_1, \boldsymbol{t}_2$.
If it is, one computes $M = [e - (\boldsymbol{u}\boldsymbol{z})]$ and outputs M. Otherwise, one outputs $\perp$.

Groth-Sahai Proof System. Groth and Sahai [GS08] proposed non-interactive zero-knowledge proofs of satisfiability of certain equations over bilinear groups, called *pairing product equations.* Using as witness group elements (and scalars) which satisfy the equation, the prover starts with making commitments on them. To prove satisfiability of an equation (which is the statement of the proof), a Groth-Sahai proof uses these commitments and shows that the committed values satisfy the equation. The proof consists again of group elements and is verified by a pairing equation derived from the statement.

We refer to [GS08] for details of the Groth-Sahai proof system, and to [EHK+13] for the compatibility with the k-MDDH assumptions. More details can be found in Appendix B. We are going to give a rough idea of the technique for SXDH.

To prove that committed variables satisfy a set of relations, the Groth-Sahai techniques require one commitment per variable and one proof element (made of a constant number of group elements) per relation. Such proofs are available for pairing-product relations and for multi-exponentiation equations.

When based on the SXDH assumption, the commitment key is of the form $\mathbf{u}_1 = (u_{1,1}, u_{1,2}), \mathbf{u}_2 = (u_{2,1}, u_{2,2}) \in \mathbb{G}_1^2$ and $\mathbf{v}_1 = (v_{1,1}, v_{1,2}), \mathbf{v}_2 = (v_{2,1}, v_{2,2}) \in \mathbb{G}_2^2$. We write

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} u_{1,1} & u_{1,2} \\ u_{2,1} & u_{2,2} \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} v_{1,1} & v_{1,2} \\ v_{2,1} & v_{2,2} \end{pmatrix}.$$

The Setup algorithm initializes the parameters as follows: $\mathbf{u}_1 = (g_1, u)$ with $u = g_1^\lambda$ and $\mathbf{u}_2 = \mathbf{u}_1{}^\mu$ with $\lambda, \mu \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, which means that $\mathbf{u}$ is a Diffie-Hellman tuple in $\mathbb{G}_1$, since $\mathbf{u}_1 = (g_1, g_1^\lambda)$ and $\mathbf{u}_2 = (g_1^\mu, g_1^{\lambda\mu})$. The TSetup algorithm will use instead $\mathbf{u}_2 = \mathbf{u}_1{}^\mu \odot (1, g_1)^{-1}$: $\mathbf{u}_1 = (g_1, g_1^\lambda)$ and $\mathbf{u}_2 = (g_1^\mu, g_1^{\lambda\mu-1})$. And it is the same in $\mathbb{G}_2$ for $\mathbf{v}$. Depending on the definition of $\mathbf{u}_2$, $\mathbf{v}_2$, this commitment can be either perfectly hiding or perfectly binding. The two parameter initializations are indistinguishable under the SXDH assumption.

To commit to $X \in \mathbb{G}_1$, one chooses randomness $s_1, s_2 \in \mathbb{Z}_p$ and sets $\mathcal{C}(X) = (1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} = (1, X) \odot (u_{1,1}^{s_1}, u_{1,2}^{s_1}) \odot (u_{2,1}^{s_2}, u_{2,2}^{s_2}) = (u_{1,1}^{s_1} \cdot u_{2,1}^{s_2}, X \cdot u_{1,2}^{s_1} \cdot u_{2,2}^{s_2})$. Similarly, one can commit to element in $\mathbb{G}_2$ and scalars in $\mathbb{Z}_p$. The committed group elements can be extracted if $\mathbf{u}_2$ is linearly dependant of $\mathbf{u}_1$ by knowing the discrete logarithm x_1 between $\mathbf{u}_{1,1}$ and $\mathbf{u}_{2,2}$: $c_2/(c_1^{x_1}) = X$.

In the following we are going to focus on proof of linear multi-scalar exponentiation in $\mathbb{G}_1$, that is to say we are going to prove equations of the form $\prod_i A_i^{y_i} = A$ where A_i are public elements in $\mathbb{G}_1$ and y_i are going to be scalars committed into $\mathbb{G}_2$.

2.4 Protocols

UC Framework. The goal of this simulation-based model [Can01] is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. Due to lack of space, a short introduction to the UC framework is given in Appendix C.

Oblivious Transfer and Password-Authenticated Key-Exchange. The security properties for these two protocols are given in terms of ideal functionalities in Appendix C.

3 Structure-Preserving Smooth Projective Hashing

3.1 Definition

In this section, we are now going to narrow the classical definition of Smooth Projective Hash Functions to what we are going to name Structure-Preserving Smooth Projective Hash Functions, in which both words, witnesses and projection keys are group elements.

Since witnesses now become group elements, this allows a full compatibility with Groth and Sahai methodology [GS08], such that for instance possessing a Non-Interactive Zero-Knowledge Proof of Knowledge can become new witnesses of our SP-SPHF, leading to interesting applications, as described later on.

As we are in the context of Structure Preserving cryptography, we assume the existence of a (prime order) bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, g_1, g_2, \mathbb{G}_T, e)$, and consider Languages (sets of elements) $\mathfrak{L}$ defined over this group. The hash space is usually $\mathbb{G}_T$, the projection key space a group $\mathbb{G}_1^m \times \mathbb{G}_2^n$ and the witness space a group $\mathbb{G}_1^n \times \mathbb{G}_2^m$.

Definition 7 (Structure-Preserving Smooth Projective Hash Functions).
A Structure-Preserving Smooth Projective Hash Function over a language $\mathfrak{L} \subset X$ onto a set $\mathcal{H}$ is defined by 4 algorithms (HashKG, ProjKG, Hash, ProjHash)*:*

- HashKG($\mathfrak{L}$, param)*, outputs a hashing key* hk *for the language $\mathfrak{L}$;*
- ProjKG(hk, ($\mathfrak{L}$, param), W)*, derives the projection key* hp *thanks to the hashing key* hk*.*
- Hash(hk, ($\mathfrak{L}$, param), W)*, outputs a hash value $H \in \mathcal{H}$, thanks to the hashing key* hk*, and W*
- ProjHash(hp, ($\mathfrak{L}$, param), W, w)*, outputs the value $H' \in \mathcal{H}$, thanks to* hp *and the witness w that $W \in \mathfrak{L}$.*

Remark 8. We stress that, contrarily to classical SPHF*, both* hp*, W and more importantly w are base group elements, and so live in the same space.*

3.2 Properties

Properties are then inherited by those of classical Smooth Projective Hash Functions.

- *Correctness*: On honest computations with (W, w) compatible with $\mathfrak{L}$, we have
$$\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w) = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W).$$
- *Smoothness*: For all $W \in X \setminus \mathfrak{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), \\ v = \mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W) \in \mathbb{G}_T \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathfrak{L}, \mathsf{param}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{param} = \mathsf{Setup}(1^{\mathfrak{K}}), \mathsf{hk} = \mathsf{HashKG}(\mathfrak{L}, \mathsf{param}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W), v \xleftarrow{\$} \mathbb{G}_T \end{array} \right\}.$$

This is formalized by

$$\mathsf{Adv}^{\mathsf{smooth}}_{\mathsf{SPHF}}(\mathfrak{K}) = \sum_{V \in \mathbb{G}} \left| \Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V] \right| \text{ is negligible.}$$

As usual, a derivative property called *Pseudo-Randomnness*, says the previous distribution are computationally indistinguishable from words in the language while the witnesses remain unknown. This is implied by the Smoothness on Hard Subset membership languages.

3.3 Retro-compatibility

Constructing SP-SPHF is not that hard of a task. A first naive approach allows to transform every pairing-less SPHF into a SP-SPHF in a bilinear setting. It should be noted that while the resulting Hash/ProjHash values live in the target group, nearly all use cases encourage to use a proper hash function on them before computing anything using their value, hence the communication cost would remain the same. (Only applications where one of the party has to provide an additional proof that the ProjHash was honestly computed might be lost, but besides proof of negativity from [BCV15], this never arises.)

To this goal, simply given a new generator $f \in \mathbb{G}_2$, and a scalar witness vector λ, one generates the new witness vector $\Lambda = [f \odot \boldsymbol{\lambda}]_2$. Words and projection keys belong to $\mathbb{G}_1$, and hash values to $\mathbb{G}_T$. Any SPHF can thus be transformed into an SP-SPHF in the following way:

	SPHF	SP-SPHF
Word $\boldsymbol{u}$	$[\boldsymbol{\lambda} \odot \Gamma(\boldsymbol{u})]_1$	$[\boldsymbol{\lambda} \odot \Gamma(\boldsymbol{u})]_1$
Witness w	$\boldsymbol{\lambda}$	$\boldsymbol{\Lambda} = [f \odot \boldsymbol{\lambda}]_2$
hk	$\boldsymbol{\alpha}$	$\boldsymbol{\alpha}$
$\mathsf{hp} = [\boldsymbol{\gamma}(\boldsymbol{u})]_1$	$[\Gamma(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_1$	$[\Gamma(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_1$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$[\Theta(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_1$	$[f \odot \Theta(\boldsymbol{u}) \odot \boldsymbol{\alpha}]_T$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	$[\boldsymbol{\lambda} \odot \boldsymbol{\gamma}(\boldsymbol{u})]_1$	$[\boldsymbol{\Lambda} \odot \boldsymbol{\gamma}(\boldsymbol{u})]_T$

- *Correctness* is inherited for words in $\mathcal{L}$ as this reduces to computing the same values but in $\mathbb{G}_T$.
- *Smoothness*: For words outside the language, the projection keys, remaining unchanged, do not reveal new information, so that the smoothness will remain preserved.
- *Pseudo-Randomness*: Without any witness, words inside the language are indistinguishable from words outside the language (under the subgroup decision assumption), hence the hash values remain pseudo-random.

It should be noted that in case this does not weaken the subgroup decision assumption (k-MDDH in the following) linked to the original language, one can set $\mathbb{G}_1 = \mathbb{G}_2$.

We give in Figure 1 two examples of regular Smooth Projective Hash Functions on Diffie-Hellman and Cramer-Shoup encryption of M, where $\alpha = \mathcal{H}(\mathbf{u}, e)$, and their counterparts with SP-SPHF. ElGamal being a simplification of Cramer-Shoup, we skip the description of the associated SP-SPHF. We also give in Figure 2 the matricial version of Cramer-Shoup encryption, in which we denote by C' the Cramer-Shoup encryption C of M in which we removed M.

3.4 Possible Applications

Nearly Constant 1-out-of-m Oblivious Transfer Using FLM. Recent pairing-based constructions [CKWZ13, ABB+13] of Oblivious Transfer use SPHF to mask each line of a database with the hash value of as SPHF on the language corresponding to the first flow being a commitment of the said line.

	SPHF	SP-SPHF
DH	h^r, g^r	h^r, g^r
Witness w	r	g_2^r
hk	λ, μ	λ, μ
hp	$h^\lambda g^\mu$	$h^\lambda g^\mu$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$(h^r)^\lambda (g^r)^\mu$	$e((h^r)^\lambda (g^r)^\mu, g_2)$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	hp^r	$e(\mathsf{hp}, g_2^r)$
CS(M;r)	$h^r M, f^r, g^r, (cd^\alpha)^r$	$h^r M, f^r, g^r, (cd^\alpha)^r$
Witness w	r	g_2^r
hk	$\lambda_1, \lambda_2, \mu, \nu, \eta$	$\lambda_1, \lambda_2, \mu, \nu, \eta$
hp	$h^{\lambda_1} f^\mu g^\nu c^\eta, h^{\lambda_2} d^\nu$	$h^{\lambda_1} f^\mu g^\nu c^\eta, h^{\lambda_2} d^\nu$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$H = (h^r)^{\lambda_1 + \alpha\lambda_2} (f^r)^\mu (g^r)^\nu ((cd^\alpha)^r)^\mu$	$e(H, g_2)$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	$(\mathsf{hp}_1 \mathsf{hp}_2^\alpha)^r$	$e(\mathsf{hp}_1 \mathsf{hp}_2^\alpha, g_2^r)$
(with $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$)		

Fig. 1. Example of conversion of classical SPHF into SP-SPHF

	SPHF	SP-SPHF
CS(M;r)	$[hr + M, \boldsymbol{A}r, (c + d\alpha)r]$	$[hr + M, \boldsymbol{A}r, (c + d\alpha)r]_1$
$\boldsymbol{B} : \begin{pmatrix} h \\ f \\ g \\ c \end{pmatrix}$	$\left[\boldsymbol{B}r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ d \end{pmatrix} \alpha r + \begin{pmatrix} M \\ 0 \\ 0 \\ 0 \end{pmatrix}\right]$	$\left[\boldsymbol{B}r + \begin{pmatrix} 0 \\ 0 \\ 0 \\ d \end{pmatrix} \alpha r + \begin{pmatrix} M \\ 0 \\ 0 \\ 0 \end{pmatrix}\right]_1$
Witness w	r	$[r]_2$
hk	$\lambda_1, \lambda_2, \mu, \nu, \eta$	$\lambda_1, \lambda_2, \mu, \nu, \eta$
hp	$\left[\mathsf{hp}_1 = \begin{pmatrix} \lambda_1 & \mu & \nu & \eta \end{pmatrix} \boldsymbol{B}\right]$, $\left[\mathsf{hp}_2 = \begin{pmatrix} \lambda_2 & 0 & 0 & \eta \end{pmatrix} \begin{pmatrix} h \\ 0 \\ 0 \\ d \end{pmatrix}\right]$	$[\mathsf{hp}_1]_1, [\mathsf{hp}_2]_1$
$\mathsf{Hash}(\mathsf{hk}, \boldsymbol{u})$	$\left[\begin{pmatrix} \lambda_1 + \alpha\lambda_2 & \mu & \nu & \eta \end{pmatrix} (C')\right]$	$\left[\begin{pmatrix} \lambda_1 + \alpha\lambda_2 & \mu & \nu & \eta \end{pmatrix} (C')\right]_T$
$\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{u}, w)$	$[(\mathsf{hp}_1 + \alpha \mathsf{hp}_2) r]$	$[(\mathsf{hp}_1 + \alpha \mathsf{hp}_2) r]_T$

Fig. 2. Example of conversion of SPHF into SP-SPHF (matricial notations)

Sadly, those constructions require special UC commitment on scalars, with equivocation and extraction capacities, leading to very inefficient constructions. In 2011, [FLM11] proposed a UC commitment, whose decommitment operation is done via group elements. In section 5, we are going to show how to combine the existing constructions with this efficient commitment using SP-SPHF, in order to obtain a very efficient round-optimal where there is no longer a growing overhead due to the commitment. As a side result, we show how to generalize the FLM commitment to any MDDH assumption.

Round-Optimal Password Authenticated Key Exchange with Adaptive Corruptions. Recent developments around SPHF-based PAKE have either lead to Round-Optimal PAKE in the BPR model [BPR00], or with static corruptions [KV11,BBC$^+$13b]. In order to achieve round-optimality, [ABB$^+$13] needs to do a bit-per-bit commitment of the password, inducing a communication cost proportional to the maximum password length.

In the following, we show how to take advantage of the SP-SPHF constructed on the FLM commitment to propose a One-Round PAKE UC secure against adaptive adversaries, providing a constant communication cost.

Using a ZKPK as a witness, Anonymous Credentials. Previous applications allow more efficient instantiations of protocols already using scalar-based SPHF. However, one can imagine additional scenarios, where a scalar based approach may not be possible, due to the inherent nature of the witness used.

For example, one should consider a strong authentication scenario, in which each user possesses an identifier delivered by an authority, and a certification on a commitment to this identifier, together with a proof of knowledge that this commitment is indeed a commitment to this identifier. (Such scenario can be transposed to the delivery of a Social Security Number, where a standalone SSN may not be that useful, but a SSN officially linked to someone is a sensitive information that should be hidden.) In this scenario, a user who wants to access his record on a government service where he is already registered, should give the certificate, and then would use an implicit proof that this corresponds to his identifier. With our technique, the server would neither learn the certificate in the clear nor the user identifier (if he did not possess it earlier), and the user would be able to authenticate only if his certificate is indeed on his committed identifier.

In our scenario, we could even add an additional step, such that Alice does not interact directly with Bob but can instead use a pawn named Carol. She could send to Carol a commitment to the signature on her identity, prove in a black box way that it is a valid signature on an identity, and let Carol do the interaction on her behalf. For example, to allow a medical practitioner to access some subpart of her medical record concerning on ongoing treatment, in this case, Carol would need to anonymously prove to the server that she is indeed a registered medical practitioner, and that Alice has given her access to her data.

4 Encryption and Commitment Schemes Based on k-MDDH

4.1 k-MDDH Cramer-Shoup Encryption

In this paper, we supersede the previous constructions with a k-MDDH based one:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with an underlying matrix assumption using a base matrix $[\boldsymbol{A}] \in \mathbb{G}^{k+1 \times k}$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = \boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{z} \overset{\$}{\leftarrow} \mathbb{Z}_p^{k+1}$, and sets, $\boldsymbol{c} = \boldsymbol{t}_1 \boldsymbol{A} \in \mathbb{Z}_p^k, \boldsymbol{d} = \boldsymbol{t}_2 \boldsymbol{A} \in \mathbb{Z}_p^k, \boldsymbol{h} = \boldsymbol{z}\boldsymbol{A} \in \mathbb{Z}_p^k$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function).
  The encryption key is $\mathsf{ek} = ([\boldsymbol{c}], [\boldsymbol{d}], [\boldsymbol{h}], [\boldsymbol{A}], \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, [m]; \boldsymbol{r})$, for a message $M = [m] \in \mathbb{G}$ and random scalars $\boldsymbol{r} \overset{\$}{\leftarrow} \mathbb{Z}_p^k$, the ciphertext is $C = (\mathbf{u} = [\boldsymbol{A}\boldsymbol{r}]), e = [\boldsymbol{h}\boldsymbol{r} + m], v = [(\boldsymbol{c} + \boldsymbol{d} \odot \xi)\boldsymbol{r}]_1$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether v is consistent with $\boldsymbol{t}_1, \boldsymbol{t}_2$.
  If it is, one computes $M = [e - (\boldsymbol{u}\boldsymbol{z})]$ and outputs M. Otherwise, one outputs $\perp$.

Theorem 9. *The k-MDDH Cramer-Shoup Encryption is IND-CCA 2 under k-MDDH assumption and the collision resistance (universal one-wayness) of the Hash Family.*

Proof. To sketch the proof of the theorem, one should remember that the original proof articulate around three main cases noting $\ell, \mathbf{u}, e, v$ the challenge query, and $\ell', \mathbf{u}, e', v'$ the current decryption query:

- $(\ell, \mathbf{u}, e) = (\ell', \mathbf{u}', e')$ but $v \neq v'$. This will fail as v is computed to be the correct checksum, hence we can directly reject the decryption query.
- $(\ell, \mathbf{u}, e) \neq (\ell', \mathbf{u}', e')$ but $\xi = \xi'$, this is a collision on the Hash Function.
- $(\ell, \mathbf{u}, e, v) \neq (\ell, \mathbf{u}, e, v)$ and $\xi \neq \xi'$. This is the argument revolving around the 2-Universality of the Hash Proof system defined by $\boldsymbol{c}, \boldsymbol{d}$. $\boldsymbol{c}, \boldsymbol{d}$ gives $2k$ equations in $2k + 2$ variables, hence answering decryption queries always in the same span can give at most 1 more equation leaving at least 1 degree of freedom in the system. $\square$

Structure-Preserving Smooth Projective Hash Function

For ease of readability we are going to set $\boldsymbol{B} = \left[\begin{pmatrix} h \\ \boldsymbol{A} \\ \boldsymbol{c} \end{pmatrix}\right]$ and $\boldsymbol{D} = \left[\begin{pmatrix} 0 \\ \vdots \\ \boldsymbol{d} \end{pmatrix}\right]$, and write $C' = [\boldsymbol{Br} + \xi \boldsymbol{Dr}]_1$ the ciphertext without the message M.

– $\mathsf{HashKG}(\mathfrak{L}, \mathsf{param})$, chooses $\Lambda \xleftarrow{\$} \mathbb{Z}_p^{(k+2)\times 1}$, $\lambda \xleftarrow{\$} \mathbb{Z}_p$ and sets

$$\mathsf{hk}_1 = \Lambda, \mathsf{hk}_2 = \begin{pmatrix} \lambda \\ \mathbf{0} \\ \Lambda_{k+2} \end{pmatrix};$$

– $\mathsf{ProjKG}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs $\mathsf{hp}_1 = \mathsf{hk}_1^\top \boldsymbol{B}, \mathsf{hp}_2 = \mathsf{hk}_2^\top \begin{pmatrix} h \\ \mathbf{0} \\ d \end{pmatrix}$;

– $\mathsf{Hash}(\mathsf{hk}, (\mathfrak{L}, \mathsf{param}), W)$, outputs a hash value $H = [(\mathsf{hk}_1 + \xi \mathsf{hk}_2)^\top C']_T$;

– $\mathsf{ProjHash}(\mathsf{hp}, (\mathfrak{L}, \mathsf{param}), W, w)$, outputs the value $H' = [(\mathsf{hp}_1 + \xi \mathsf{hp}_2)\boldsymbol{r}]_T$.

The Smoothness comes inherently from the fact that we have $2k+2$ unknowns in hk while hp gives at most $2k$ equations. Hence an adversary has a negligible chance to find the real values.

4.2 A Universally Composable Commitment with Adaptive Security Based on MDDH

We first show how to simply generalize FLM's commitment [FLM11] from DLin to **k**-MDDH.

FLM's Commitment on DLin. At Asiacrypt 2011, Fischlin, Libert and Manulis presented a universally composable commitment [FLM11] with adaptive security based on the Decision Linear assumption [BBS04]. We show here how to generalize their scheme to the Matrix Decisional Diffie-Hellman assumption from [EHK+13] and recalled in Section 2. We first start by recalling their original scheme. Note that sid denotes the session identifier and cid the commitment identifier and that the combination $(\mathsf{sid}, \mathsf{cid})$ is globally unique, as in [HMQ04, FLM11].

– **CRS Generation:** $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T)$ of order $p > 2^{\mathfrak{K}}$, a generator g of $\mathbb{G}$, and sets $g_1 = g^{\alpha_1}$ and $g_2 = g^{\alpha_2}$ with random $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$. It defines the vectors $\mathbf{g_1} = (g_1, 1, g)$, $\mathbf{g_2} = (1, g_2, g)$ and $\mathbf{g_3} = \mathbf{g_1}^{\xi_1} \mathbf{g_2}^{\xi_2}$ with random $\xi_1, \xi_2 \in \mathbb{Z}_p^*$, which form a Groth-Sahai CRS $\mathbf{g} = (\mathbf{g_1}, \mathbf{g_2}, \mathbf{g_3})$ for the perfect soundness setting. It then chooses a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p$ and generates a public key $\mathsf{pk} = (X_1, \ldots, X_6)$ for the linear Cramer-Shoup encryption scheme. The CRS consists of $\mathsf{crs} = (\mathfrak{K}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{g}, H, \mathsf{pk})$.

– **Commitment algorithm:** $\mathsf{Com}(\mathsf{crs}, M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$, to commit to message $M \in \mathbb{G}$ for party P_j, party P_i parses crs as $(\mathfrak{K}, \mathbb{G}, \mathbb{G}_T, g, \mathbf{g}, H, \mathsf{pk})$ and conducts the following steps:
  - It chooses random exponents r, s in $\mathbb{Z}_p$ and computes a linear Cramer- Shoup encryption $\psi_{CS} = (U_1, U_2, U_3, U_4, U_5)$ of $M \in \mathbb{G}$ under the label $\ell = P_i \| \mathsf{sid} \| \mathsf{cid}$ and the public key pk.
  - It generates a NIZK proof $\pi_{val-enc}$ that $\psi_{CS} = (U_1, U_2, U_3, U_4, U_5)$ is indeed a valid encryption of $M \in \mathbb{G}$. This requires to commit to exponents r, s and prove that these exponents satisfy the multi-exponentiation equations $U_1 = {g_1}^r$, $U_2 = {g_2}^s$, $U_3 = g^{r+s}$, $U_4/M = {X_5}^r {X_6}^s$ and $U_5 = (X_1 {X_3}^\alpha)^r \cdot (X_2 {X_4}^\alpha)^s$.
  - P_i erases (r, s) after the generation of $\pi_{val-enc}$ but retains the $D_M = \pi_{val-enc}$.

  The commitment is ψ_{CS}.

– **Verification algorithm:** the algorithm $\mathsf{VerCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ checks the proof $\pi_{val-enc}$ and ignores the opening if the verification fails.

– **Opening algorithm:** $\mathsf{OpenCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ reveals M and $D_M = \pi_{val-enc}$ to P_j.

The extraction algorithm uses Cramer-Shoup decryption algorithm, while the equivocation uses the simulator of the NIZK. It is shown in [ABB$^+$13] that the IND-CCA security notion for C and the computational soundness of π make it strongly-binding-extractable, while the IND-CCA security notion and the zero-knowledge property of the NIZK provide the strong-simulation-indistinguishability.

Moving to k-MDDH: We now show how to extend the previous commitment to the k-MDDH assumption. Compared to the original version of the commitment, we split the proof $\pi_{val-enc}$ into its two parts: the NIZK proof denoted here as $[\boldsymbol{\Pi}]_1$ is still revealed during the opening algorithm, while the Groth-Sahai commitment $[\boldsymbol{R}]_2$ of the randomness $\mathbf{r}$ of the Cramer-Shoup encryption is sent during the commitment phase. Furthermore, since the hash value in the Cramer Shoup encryption is used to link the commitment with the session, we include this value $[\boldsymbol{R}]_2$ to the label, in order to ensure that this extra commitment information given with the ciphertext is the original one. We refer the reader to the original security proof in [FLM11, Theorem 1], which remains exactly the same, since this additional commitment provides no information (either computationally or perfectly, depending on the CRS), and since the commitment $[\boldsymbol{R}]_2$ is not modified in the equivocation step (only the value $[\boldsymbol{\Pi}]_1$ is changed).

- **CRS Generation:** algorithm $\mathsf{SetupCom}(1^{\mathfrak{K}})$ chooses a bilinear asymmetric group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ of order $p > 2^{\mathfrak{K}}$, and a set of generators $[\boldsymbol{A}]_1$ corresponding to the underlying matrix assumption.
  As explained in [EHK$^+$13], following their notations, one can define a Groth-Sahai CRS by picking $\boldsymbol{w} \overset{\$}{\leftarrow} \mathbb{Z}_p^{k+1}$, and setting $[\boldsymbol{U}]_2 = [\boldsymbol{B}||\boldsymbol{B}\boldsymbol{w}]_2$ for a hiding CRS, and $[\boldsymbol{B}||\boldsymbol{B}\boldsymbol{w} + (0||z)^\top]_2$ otherwise, where $[\boldsymbol{B}]_2$ is an k-MDDH basis, and $\boldsymbol{w}, z$ are the elements defining the challenge vector.
  For the Cramer-Shoup like CCA-2 encryption, one additionally picks $\boldsymbol{t}_1, \boldsymbol{t}_2, \boldsymbol{z} \overset{\$}{\leftarrow} \mathbb{Z}_p^{k+1}$, and a Universal One-Way Hash Function $\mathcal{H}$ and sets $[\boldsymbol{h}]_1 = [\boldsymbol{z} \cdot \boldsymbol{A}]_1$, $[\boldsymbol{c}]_1 = [\boldsymbol{t}_1\boldsymbol{A}]_1, [\boldsymbol{d}]_1 = [\boldsymbol{t}_2\boldsymbol{A}]_1$.
  The CRS consists of $\mathsf{crs} = (\mathfrak{K}, p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, [\boldsymbol{A}]_1 \in \mathbb{G}_1^{k \times k+1}, [\boldsymbol{U}]_2, [\boldsymbol{h}]_1 \in \mathbb{G}_1^k, [\boldsymbol{c}]_1 \in \mathbb{G}_1^k, [\boldsymbol{d}]_1 \in \mathbb{G}_1^k, \mathcal{H})$.
- **Commitment algorithm:** $\mathsf{Com}(\mathsf{crs}, M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$, to commit to message $M \in \mathbb{G}_1$ for party P_j, party P_i conducts the following steps:
  - It chooses random exponents $\boldsymbol{r}$ in $\mathbb{Z}_p^k$ and commits to $\boldsymbol{r}$ in $[\boldsymbol{R}]_2$ with randomness $\rho \overset{\$}{\leftarrow} \mathbb{Z}_p^{k \times k+1}$, setting $[\boldsymbol{R}]_2 = [\boldsymbol{U}\boldsymbol{\rho} + \iota_2(\boldsymbol{r})]_2 \in \mathbb{G}_2^{k \times k+1}$. It also computes a Cramer-Shoup encryption $\psi_{CS} = [\boldsymbol{C}]_1$ of $M \in \mathbb{G}_1$ under the label $\ell = P_i||\mathsf{sid}||\mathsf{cid}$ and the public key pk:
  $$[\boldsymbol{C}]_1 = [\boldsymbol{A}\boldsymbol{r}||\boldsymbol{h}\boldsymbol{r} + M||(\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell||\boldsymbol{C}_1||\boldsymbol{C}_2||\boldsymbol{R}))\boldsymbol{r}]_1 = [\boldsymbol{C}_1||\boldsymbol{C}_2||\boldsymbol{C}_3]_1$$
  For simplicity we write $\ell' = \ell||[\boldsymbol{C}_1]_1||[\boldsymbol{C}_2]_1||[\boldsymbol{R}]_2$.
  - It generates a NIZK proof $D_M = [\boldsymbol{\Pi}]_1$ that ψ_{CS} is indeed a valid encryption of $M \in \mathbb{G}_1$ for the committed $\boldsymbol{r}$ in $[\boldsymbol{R}]_2$. This requires to prove that these exponents satisfy the multi-exponentiation equations:
  $$[\boldsymbol{C}_1]_1 = [\boldsymbol{A}\boldsymbol{r}]_1, [\boldsymbol{C}_2 - M]_1 = [\boldsymbol{h}\boldsymbol{r}]_1, [\boldsymbol{C}_3 = (\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell'))\boldsymbol{r}]_1$$
  The associated proof is then $[\boldsymbol{\Pi}]_1 = [\boldsymbol{\rho}^\top(\boldsymbol{A}||\boldsymbol{h}||\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell'))]_1$.
  - P_i erases r after the generation of $[\boldsymbol{R}]_2$ and $[\boldsymbol{\Pi}]_1$ but retains $D_M = [\boldsymbol{\Pi}]_1$.

  The commitment is $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$.
- **Verification algorithm:** the algorithm $\mathsf{VerCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ checks the consistency of the proof $\pi_{val-enc}$ with respect to $[\boldsymbol{C}]_1$ and $[\boldsymbol{R}]_2$. and ignores the opening if the verification fails.
- **Opening algorithm:** $\mathsf{OpenCom}(\mathsf{crs}, M, D_M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ reveals M and $D_M = [\Pi]_1$ to P_j.

One can easily see that $[\boldsymbol{C}_3]_1$ is the projective hash computation of a 2-universal hash proof on the language "$[\boldsymbol{C}_1]_1$ in the span of $\boldsymbol{A}$", with $[\boldsymbol{C}_2]_1$ being an additional term that uses the same witness to mask the committed message, so that $[\boldsymbol{C}]_1$ is a proper generalization of the Cramer-Shoup CCA-2 encryption. Details on the k-MDDH Groth-Sahai proofs are given in Appendix B.

It is thus easy to see that this commitment is indeed a generalization of the FLM non-interactive UC commitment with adaptive corruption under reliable erasures (in which we switched the CRS, the Cramer-Shoup encryption and the Groth-Sahai proof in the k-MDDH setting).

4.3 A Structure-Preserving Smooth Projective Hash Function Associated with this Commitment

Structure-Preserving Smooth Projective Hash Function. We now want to supersede the verification equation of the commitment by a smooth projective hash function providing implicit decommitment, simply using the proof as a witness. We consider the language of the valid encryptions of M using a random r which is committed into $[\boldsymbol{R}]_2$:

$$\begin{aligned}\mathcal{L}_M = \{[\boldsymbol{C}]_1 \mid \exists r \exists \rho \text{ such that } & [\boldsymbol{R}]_2 = [\boldsymbol{U\rho} + \iota_2(\boldsymbol{r})]_2 \\ \text{and } & [\boldsymbol{C}]_1 = [\boldsymbol{Ar} || \boldsymbol{hr} + M || (\boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell || \boldsymbol{C}_1 || \boldsymbol{C}_2 || \boldsymbol{R}))\boldsymbol{r}]_1\}\end{aligned}$$

The verifier picks a random $\mathsf{hk} = \alpha \xleftarrow{\$} \mathbb{Z}_p^{k+3 \times k+1}$ and sets $\mathsf{hp} = [\alpha \odot \boldsymbol{U}]_2$.
On one side, the verifier then computes:

$$\mathsf{Hash}(\mathsf{hk}, ([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)) = [\alpha \odot ((\boldsymbol{C}_1 || \boldsymbol{C}_2 - M || \boldsymbol{C}_3) - (\boldsymbol{A} || \boldsymbol{h} || \boldsymbol{c} + \boldsymbol{d} \odot \mathcal{H}(\ell')) \cdot \boldsymbol{R})]_T$$

While the prover computes $\mathsf{ProjHash}(\mathsf{hp}, \boldsymbol{\Pi}) = [\boldsymbol{\Pi} \cdot \mathsf{hp}]_T$.

– *Correctness*: comes directly from the previous equations.
– *Smoothness*: on a binding CRS, $[\boldsymbol{U}]_2$'s last column is in the span of the k first (which are simply $[\boldsymbol{B}]_2$), hence as $\mathsf{hk} \in \mathbb{Z}_p^{k+1}$, the k equations given in hp are not enough to determine its value and so it is still perfectly hidden from an information theoretic point of view.
– *Pseudo-Randomness*: Under the MDDH assumption, the subset membership decision is a hard problem, as the generalized Cramer-Shoup is IND-CCA-2, and $[\boldsymbol{R}]_2$ is an IND-CPA commitment to $\boldsymbol{r}$.

Theorem 10. *Under the k-MDDH assumption, the above SP-SPHF is strongly pseudo-random on a perfectly hiding CRS.*

For sake of compactness, the proof is postponed to Appendix D.

Efficiency. The rough size of a projection key is $k \times (k+3)$ (number of elements in each proof times number of proofs). It should be noted, that for a CS-SPHF (in the case of the oblivious transfer), instead of repeating the projection key $k+3$ times (in order to verify each component of the Cramer-Shoup), one can generate a value $\varepsilon \xleftarrow{\$} \mathbb{Z}_p$, an hp for a single equation, and say that for the other component, one simply uses $\mathsf{hp}^{\varepsilon^i}$, as the trick explained in [ABB+13].

5 Application: Nearly Optimal Size 1-out-of-m Oblivious Transfer

5.1 Main Idea of the Construction

Our oblivious transfer scheme builds upon that presented by Abdalla *et al.* at Asiacrypt 2013 [ABB+13]. In their scheme, the authors use a SPHF-friendly commitment (which is a notion stronger than a UC commitment) along with its associated SPHF in a now classical way to implicitly open the commitment. They claim that the commitment presented in [FLM11] cannot be used in such an application, since it is not "robust", which is a security notion meaning that one cannot produce a commitment and a label that extracts to x' (possibly $x' = \perp$) such that there exists a valid opening data to a different input x, even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom). Indeed, because of the perfectly-hiding setting of Groth-Sahai proofs, for any ciphertext C and for any message x, there exists a proof Π that makes the verification of C on x. However, we show in this section that in spite of this result, such a commitment can indeed be used in a relatively close construction of oblivious transfer scheme.

To this aim, we use our construction of structure-preserving SPHF on FLM's commitment, simply using the decommitment value (a Groth-Sahai proof) as the witness, presented in Section 4.3.

It should be noted that the commitment used in [ACP09, ABB$^+$13] has the major drawback of leaking the bit-length of the committed message. While in application to Oblivious Transfer this is not a major problem, for PAKE this is a way more sensitive issue, as we show in the next section. Moreover, using FLM's commitment is conceptually simpler, since the equivocation only needs to modify the witness, allowing the user to compute honestly its message in the commitment phase, whereas in the original commitments, a specific flow had to be sent during the commitment phase (with a different computation and more witnesses for the SPHF, than in the honest computation of the commitment).

5.2 A Universally Composable Oblivious Transfer with Adaptive Security Based on MDDH

We denote by DB the database of the server containing $t = 2^m$ lines, and j the line requested by the user in an oblivious way. We assume the existence of a Pseudo-Random Generator (PRG) F with input size equal to the plaintext size, and output size equal to the size of the messages in the database and a IND-CPA encryption scheme $\mathcal{E} = (\mathsf{Setup}_{\mathrm{cpa}}, \mathsf{KeyGen}_{\mathrm{cpa}}, \mathsf{Encrypt}_{\mathrm{cpa}}, \mathsf{Decrypt}_{\mathrm{cpa}})$ with plaintext size at least equal to the security parameter. The commitment used is the variant of [FLM11] described above. It is denoted as Com^ℓ in the description of the scheme, with ℓ being a label. Note that sid denotes the session identifier, ssid the subsession identifier and cid the commitment identifier and that the combination (sid, cid) is globally unique, as in [HMQ04, FLM11].

We present our construction, in Figure 3, following the global framework presented in [ABB$^+$13], for an easier efficiency comparison (we achieve nearly optimality in the sense that it is linear in the number of lines of the database, but with a constant equal to 1 only).

CRS generation:
$\mathsf{crs} \xleftarrow{\$} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \xleftarrow{\$} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow:

1. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen
2. Server sends pk to User

Index query on j:

1. User chooses a random value J, computes $S \leftarrow F(J)$ and encrypts J under pk: $c \xleftarrow{\$} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. User computes $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \xleftarrow{\$} \mathsf{Com}^\ell(\mathsf{crs}, j, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with the label $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
3. User stores $[\boldsymbol{\Pi}]_1$ and completely erases J and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $[\boldsymbol{C}]_1, [\boldsymbol{R}]_2$ and c to Server

Database input $(n_1, \ldots, n_t)$:

1. Server decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $S \leftarrow F(J)$
2. For $s = 1, \ldots, t$: Server computes $\mathsf{hk}_s \xleftarrow{\$} \mathsf{HashKG}(\mathcal{L}_s)$, $\mathsf{hp}_s \leftarrow \mathsf{ProjKG}(\mathsf{hk}_s, \mathcal{L}_s)$, $K_s \leftarrow \mathsf{Hash}(\mathsf{hk}_s, (\mathcal{L}_s, (\ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2)))$, and $N_s \leftarrow S \oplus K_s \oplus n_s$
3. Server erases everything except $(\mathsf{hp}_s, N_s)_{s=1,\ldots,t}$ and sends them over a secure channel

Data recovery:
Upon receiving $(\mathsf{hp}_s, N_s)_{s=1,\ldots,t}$, User computes
$K_j \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathcal{L}_j, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}]_1)$ and gets $n_j \leftarrow S \oplus K_j \oplus N_j$.

Fig. 3. UC-Secure 1-out-of-t OT from an SPHF-Friendly Commitment (for Adaptive Security)

Theorem 11. *The oblivious transfer scheme described in Figure 3 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

The proof is given in Appendix E for completeness.

6 Application: Adaptive and Length-Independent One-Round PAKE

Password-authenticated key exchange (PAKE) protocols allow two players to agree on a shared high entropy secret key, that depends on their own passwords only. Katz and Vaikuntanathan recently came up with the first concrete one-round PAKE protocols [KV09], where the two players just have to send simultaneous flows to each other. Following their idea, [BBC$^+$13b] proposed a round-optimal PAKE protocol UC secure against passive corruptions. On the other hand, [ACP09] proposed the first protocol UC secure against adaptive corruptions, and [ABB$^+$13] built upon both [KV09] and [ACP09], to propose the first one-round protocol UC secure against adaptive corruptions. Unfortunately, both of them share a drawback, which is that they use a commitment growing linearly with the length of a password. Besides being an efficiency problem, it is over all a security issue in the UC framework. Indeed, the simulator somehow has to "guess" the length of the password of the player it simulates, otherwise it is unable to equivocate the commitment (since the commitment reveals the length of the password it commits to). Since such a guess is impossible, the apparently only solution to get rid of this limitation seems to give the users an upper-bound on the length of their passwords and to ask them to compute commitments of this length, which leads to costly computations.

In this section, we are now going to present a constant-size, round-optimal, PAKE UC secure against adaptive corruptions. It builds upon the protocol proposed in [ABB$^+$13], using the same techniques as in the former section to avoid the apparent impossibility to use FLM's commitment.

CRS: $\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}})$.
Protocol execution by P_i with pw_i:

1. P_i generates $\mathsf{hk}_i \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathcal{L}_{\mathsf{pw}_i})$, $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathcal{L}_{\mathsf{pw}_i})$ and erases any random coins used for the generation
2. P_i computes $([\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2, [\boldsymbol{\Pi}_i]_1) = \mathsf{Com}^{\ell_i}(\mathsf{crs}, \mathsf{pw}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$
3. P_i stores $[\boldsymbol{\Pi}_i]_1$, completely erases random coins used by Com and sends $\mathsf{hp}_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2$ to P_j

Key computation: Upon receiving $\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2$ from P_j

1. P_i computes $H_i' \leftarrow \mathsf{ProjHash}(\mathsf{hp}_j, (\mathcal{L}_{\mathsf{pw}_i}, \ell_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2), [\Pi_i]_1))$ and $H_j \leftarrow \mathsf{Hash}(\mathsf{hk}_i, (\mathcal{L}_{\mathsf{pw}_i}, \ell_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2))$ with $\ell_j = (\mathsf{sid}, P_j, P_i, \mathsf{hp}_j)$
2. P_i computes $\mathsf{sk}_i = H_i' \cdot H_j$ and erases everything else, except pw_i.

Fig. 4. UC-Secure PAKE from the revisited FLM Commitment

It should be noted that we need the classical requirement for extraction capabilities (see for example [Lin11, BCPV13] for a detailed explanation), *i.e.* a password pw is assumed to be a bit-string of length bounded by $\log p - 2$, and then one can use a bijective embedding function G mapping $\{0,1\}^{|p|-2}$ in $\mathbb{G}_1$. For the sake of simplicity, we continue to write pw_i in the high level description, but it should be interpreted as a commitment to $G(\mathsf{pw}_i)$.

The language $\mathcal{L}_{\mathsf{pw}_i}$ is then the language of valid Cramer-Shoup encryptions of the embedded password $G(\mathsf{pw}_i)$, consistent with the randomness committed in the second part, and the rest of the label.

Theorem 12. *The Password Authenticated Key Exchange scheme described in Figure 4 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

The proof is given in Appendix F for completeness.

7 Application: Anonymous Credential-Based Message Transmission

Anonymous Credential protocols [Cha86, Dam90, CL01] allow to combine security and privacy. Typical credential use involves three main parties. Users need to interact with some authorities to obtain their credentials (assumed to be a set of attributes validated / signed), and then prove to a server that a subpart of their attributes verifies an expect policy.

In this section, we give another go to Anonymous Credential, this time to allow message recovery. This is between Anonymous Credential but also Conditional Oblivious Transfer [Rab81] and Oblivious Signature-Based Envelope [LDB03].

We present a constant-size, round-optimal protocol that allow to use a Credential to retrieve a message without revealing the Anonymous Credentials in a UC secure way, by simply building on the commitment proposed earlier in the paper.

7.1 Anonymous Credential System

In a Attribute-Based Credential system, we assume that different organization issue credentials to users. A user i possesses a set of credential Cred_i of the form $\{\mathsf{Cred}_{i,j}, \mathsf{vk}_j\}$ where organization j assesses that the user verifies some property. (The DMV will assess that the user is indeed capable of driving, the university that she has a bachelor in Computer Science, while Squirrel Airways that she reached the gold membership, all those authorities don't communicate with each other).

A Server might have an access Policy P requiring some elements (For example being a female, with a bachelor, and capable of driving).

- $\mathsf{Setup}(1^{\mathfrak{K}})$: A probabilistic algorithm that gets a security parameter $\mathfrak{K}$, an upper bound t for the size of attribute sets and returns the public parameters param
- $\mathsf{OKeyGen}(\mathsf{param})$: Generates a pair of signing keys $\mathsf{sk}_j, \mathsf{vk}_j$ for each organization.
- $\mathsf{UKeyGen}(\mathsf{param})$: Generates a pair of keys $\mathsf{sk}_i, \mathsf{vk}_i$ for each use.
- $\mathsf{CredObtain}(\langle U_i, \mathsf{sk}_i\rangle, \langle O_j, \mathsf{sk}_j\rangle)$ Interactive process that allows a user i to obtain some credentials from organization j by providing his public key vk_j and a proof that it belongs to him.
- $\mathsf{CredUse}(\langle U_i, \mathsf{Cred}_i, \mathsf{sk}_i\rangle, \langle S, P, M\rangle)$ Interactive process that allows a user i to access a message guarded by the server S under some policy P by using the already obtained credentials.

An attribute-based anonymous credential system is called secure if it is correct, unforgeable and anonymous.

7.2 Construction

Smooth Projective Hash Functions have been shown to handle complex languages [ACP09, BBC$^+$13a], those properties can naturally be extended to Structure Preserving Smooth Projective Hash Function, allowing credentials to be expressive as disjunction / conjunction of sets of credentials, range proofs, or even composition (having a credential from authority A signed by authority B for example).

What is really new with the Structure Preserving part is that now a user can request to have a credential on a witness by requiring a Structure-Preserving signature on it, while before

CRS generation:
$\mathsf{crs} \overset{\$}{\leftarrow} \mathsf{SetupCom}(1^{\mathfrak{K}}), \mathsf{param}_{\mathrm{cpa}} \overset{\$}{\leftarrow} \mathsf{Setup}_{\mathrm{cpa}}(1^{\mathfrak{K}})$.

Pre-flow:

1. Server generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}_{\mathrm{cpa}}(\mathsf{param}_{\mathrm{cpa}})$ for $\mathcal{E}$, stores sk and completely erases the random coins used by KeyGen
2. Server sends pk to User

Credential Use by user i:

1. User chooses a random value J, computes $S \leftarrow F(J)$ and encrypts J under pk: $c \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathrm{cpa}}(\mathsf{pk}, J)$
2. User computes $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(\mathsf{crs}, \mathsf{Cred}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$
3. User stores $[\boldsymbol{\Pi}]_1$ and completely erases J and the random coins used by Com and $\mathsf{Encrypt}_{\mathrm{cpa}}$ and sends $[\boldsymbol{C}]_1, [\boldsymbol{R}]_2$ and c to Server

Database input M with policy P:

1. Server decrypts $J \leftarrow \mathsf{Decrypt}_{\mathrm{cpa}}(\mathsf{sk}, c)$ and computes $S \leftarrow F(J)$
2. Server computes $\mathsf{hk}_P \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathcal{L}_P)$, $\mathsf{hp}_P \leftarrow \mathsf{ProjKG}(\mathsf{hk}_P, \mathcal{L}_P)$, $K_P \leftarrow \mathsf{Hash}(\mathsf{hk}_P, (\mathcal{L}_P, (\ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2)))$, and $N_P \leftarrow S \oplus K_P \oplus M$
3. Server erases everything except (hp_P, N_P) and sends them over a secure channel

Data recovery:
Upon receiving (hp_P, N_P), User computes
$K \leftarrow \mathsf{ProjHash}(\mathsf{hp}_P, (\mathcal{L}_P, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}]_1)$ and gets $M \leftarrow S \oplus K \oplus N_P$.

Fig. 5. UC-Secure Anonymous Credential from an SPSPHF-Friendly Commitment (for Adaptive Security)

scalars either required to give too much information to the server B or prevented chaining as most signatures requires some sort of Hashing (BLS requires an explicit Hash, while signature *à la* Waters requires to handle a bit per bit version of the message hindering drastically the efficiency of the protocol). This allows more possibilities in both the Credential Generation step and the policy required for accessing messages, while maintaining an efficient construction.

Theorem 13. *The Anonymous Credential Protocol described in Figure 5 is* UC*-secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

The ideal functionality and a sketch of the proof are given in Appendix G for completeness.

References

[ABB+13] Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, December 2013.

[ACD+12] Masayuki Abe, Melissa Chase, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Constant-size structure-preserving signatures: Generic constructions and simple assumptions. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 4–24. Springer, December 2012.

[ACHdM05] Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/2005/385`.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, August 2009.

[ADK+13] Masayuki Abe, Bernardo David, Markulf Kohlweiss, Ryo Nishimaki, and Miyako Ohkubo. Tagged one-time signatures: Tight security and optimal tag size. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 312–331. Springer, February / March 2013.

[AFG+10] Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, August 2010.

[AGHO11] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 649–666. Springer, August 2011.

[AGOT14a] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Structure-preserving signatures from type II pairings. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 390–407. Springer, August 2014.

[AGOT14b] Masayuki Abe, Jens Groth, Miyako Ohkubo, and Mehdi Tibouchi. Unified, minimal and selectively randomizable structure-preserving signatures. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 688–712. Springer, February 2014.

[BBC$^+$13a] Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 272–291. Springer, February / March 2013.

[BBC$^+$13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, August 2013.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004.

[BC15] Olivier Blazy and Céline Chevalier. Generic construction of uc-secure oblivious transfer. Cryptology ePrint Archive, Report 2015/560, 2015.

[BCL$^+$05] Boaz Barak, Ran Canetti, Yehuda Lindell, Rafael Pass, and Tal Rabin. Secure computation without authentication. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 361–377. Springer, August 2005.

[BCPV13] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Analysis and improvement of Lindell's UC-secure commitment schemes. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 534–551. Springer, June 2013.

[BCV15] Olivier Blazy, Céline Chevalier, and Damien Vergnaud. Non-interactive zero-knowledge proofs of non-membership. Cryptology ePrint Archive, Report 2015/072, 2015. `http://eprint.iacr.org/`.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

[BPR00] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 139–155. Springer, May 2000.

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, March 2012.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CF01] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, August 2001.

[Cha86] David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In Franz Pichler, editor, *EUROCRYPT'85*, volume 219 of *LNCS*, pages 241–244. Springer, April 1986.

[CHK$^+$05] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 404–421. Springer, May 2005.

[CK02] Ran Canetti and Hugo Krawczyk. Universally composable notions of key exchange and secure channels. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 337–351. Springer, April / May 2002.

[CKWZ13] Seung Geol Choi, Jonathan Katz, Hoeteck Wee, and Hong-Sheng Zhou. Efficient, adaptively secure, and composable oblivious transfer with a single, global CRS. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 73–88. Springer, February / March 2013.

[CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.

[CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *34th ACM STOC*, pages 494–503. ACM Press, May 2002.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *CRYPTO'98*, volume 1462 of *LNCS*, pages 13–25. Springer, August 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, April / May 2002.

[Dam90] Ivan Damgård. Payment systems and credential mechanisms with provable security against abuse by individuals. In Shafi Goldwasser, editor, *CRYPTO'88*, volume 403 of *LNCS*, pages 328–335. Springer, August 1990.

[EHK$^+$13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, August 2013.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, August 1984.

[FLM11] Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 468–485. Springer, December 2011.

[GL03] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008.

[HK07] Omer Horvitz and Jonathan Katz. Universally-composable two-party computation in two rounds. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *LNCS*, pages 111–129. Springer, August 2007.

[HMQ04] Dennis Hofheinz and Jörn Müller-Quade. Universally composable commitments using random oracles. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 58–76. Springer, February 2004.

[JR14] Charanjit S. Jutla and Arnab Roy. Dual-system simulation-soundness with applications to uc-pake and more. Cryptology ePrint Archive, Report 2014/805, 2014.

[JR16] Charanjit Jutla and Arnab Roy. Smooth nizk arguments with applications to asymmetric uc-pake. Cryptology ePrint Archive, Report 2016/233, 2016. `http://eprint.iacr.org/`.

[Kal05] Yael Tauman Kalai. Smooth projective hashing and two-message oblivious transfer. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 78–95. Springer, May 2005.

[KOY01] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 475–494. Springer, May 2001.

[KPW15] Eike Kiltz, Jiaxin Pan, and Hoeteck Wee. Structure-preserving signatures from standard assumptions, revisited. In *CRYPTO 2015, Part II*, LNCS, pages 275–295. Springer, August 2015.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, December 2009.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, March 2011.

[LDB03] Ninghui Li, Wenliang Du, and Dan Boneh. Oblivious signature-based envelope. In Elizabeth Borowsky and Sergio Rajsbaum, editors, *22nd ACM PODC*, pages 182–189. ACM, July 2003.

[Lin11] Yehuda Lindell. Highly-efficient universally-composable commitments based on the DDH assumption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 446–466. Springer, May 2011.

[NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In S. Rao Kosaraju, editor, *12th SODA*, pages 448–457. ACM-SIAM, January 2001.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *LNCS*, pages 554–571. Springer, August 2008.

[Rab81] Michael O. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR81, Harvard University, 1981.

[RS92] Charles Rackoff and Daniel R. Simon. Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 433–444. Springer, August 1992.

A Commitments and Smooth Projective Hash Functions

A.1 Encryption

An encryption scheme $\mathcal{C}$ is described through four algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key pk and a (private) decryption key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, M; \rho)$ outputs a ciphertext $\mathcal{C}$, on M, under the encryption key pk, with the randomness ρ;
- $\mathsf{Decrypt}(\mathsf{dk}, \mathcal{C})$ outputs the plaintext M, encrypted in the ciphertext $\mathcal{C}$ or $\perp$.

  Such encryption scheme is required to have the following security properties:
- *Correctness*: For every pair of keys $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every messages M, and every random ρ, we should have

$$\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, M; \rho)) = M.$$

- *Indistinguishability under Adaptive Chosen Ciphertext Attack* $\mathsf{IND\text{-}CCA}$ (see [NY90, RS92]): An adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two original plaintexts, and ask several decryption of ciphertexts different from challenge one.

  The $\mathsf{ODecrypt}$ oracle outputs the decryption of c under the challenge decryption key dk. The input queries (c) are added to the list $\mathcal{CT}$ of decrypted ciphertexts.

```
$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(M_0, M_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathsf{pk}, \mathsf{ODecrypt}(\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, M_b)$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot))$
6. IF $(c^*) \in \mathcal{CT}$ RETURN 0
7. ELSE RETURN $b'$
```

A.2 Commitments

A commitment scheme is said *equivocable* if it has a second setup $\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and two algorithms

- $\mathsf{SimCom}^{\ell}(\tau)$ takes as input the trapdoor τ and a label ℓ and outputs a pair (C, eqk), where C is a commitment and eqk an equivocation key;
- $\mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$ takes as input a commitment C, a label ℓ, a message x, an equivocation key eqk, and outputs an opening data δ for C and ℓ on x.

such as the following properties are satisfied: *trapdoor correctness* (all simulated commitments can be opened on any message), *setup indistinguishability* (one cannot distinguish the CRS ρ generated by $\mathsf{SetupCom}$ from the one generated by $\mathsf{SetupComT}$) and *simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to fake commitments), denoting by SCom the algorithm that takes as input the trapdoor τ, a label ℓ and a message x and which outputs $(C, \delta) \overset{\$}{\leftarrow} \mathsf{SCom}^{\ell}(\tau, x)$, computed as $(C, \mathsf{eqk}) \overset{\$}{\leftarrow} \mathsf{SimCom}^{\ell}(\tau)$ and $\delta \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, C, x)$.

A commitment scheme $\mathcal{C}$ is said to be *extractable* if it has a second setup$\mathsf{SetupComT}(1^{\mathfrak{K}})$ that additionally outputs a trapdoor τ, and a new algorithm

```
$\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-}b}(\mathfrak{K})$
    $(\rho, \tau) \xleftarrow{\$} \mathsf{SetupComT}(1^{\mathfrak{K}})$
    $(\ell, x, \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\rho)$; $C \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$
    $\mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(L_x)$; $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, L_x, \perp)$
    If $(b = 0)$ $H \leftarrow \mathsf{Hash}(\mathsf{hk}, L_x, (\ell, C))$
    Else $H \xleftarrow{\$} \Pi$
    $(\ell', C', \mathsf{state}) \xleftarrow{\$} \mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(\mathsf{state}, C, \mathsf{hp}, H)$
    If $((\ell', ?, C') \in \Lambda)$ THEN $H' \leftarrow \perp$
    Else $H' \leftarrow \mathsf{Hash}(\mathsf{hk}, L_x, (\ell', C')$
        Return $\mathcal{A}^{\mathsf{SCom}^{\cdot}(\tau,\cdot),\mathsf{ExtCom}^{\cdot}(\tau,\cdot)}(H')$
```

Fig. 6. Strong Pseudo-Randomness

– $\mathsf{ExtCom}^{\ell}(\tau, C)$ which takes as input the trapdoor τ, a commitment C, and a label ℓ, and outputs the committed message x, or $\perp$ if the commitment is invalid.

such as the following properties are satisfied: *trapdoor correctness* (all commitments honestly generated can be correctly extracted: for all ℓ, x, if $(C, \delta) \xleftarrow{\$} \mathsf{Com}^{\ell}(x)$ then $\mathsf{ExtCom}^{\ell}(C, \tau) = x$), *setup indistinguishability* (as above) and *binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data to an input x while the commitment does not extract to x).

A commitment scheme is said *extractable and equivocable* if the indistinguishable setup algorithm outputs a common trapdoor that allows both equivocability and extractability, and the following properties are satisfied: *strong simulation indistinguishability* (one cannot distinguish a real commitment (generated by Com) from a fake commitment (generated by SCom), even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom)) and *strong binding extractability* (one cannot fool the extractor, *i.e.*, produce a commitment and a valid opening data (not given by SCom) to an input x while the commitment does not extract to x, even with oracle access to the extraction oracle (ExtCom) and to fake commitments (using SCom)).

A.3 Smooth Projective Hash Functions Used With Commitments

The strong pseudo-randomness property, taken from [BBC+13b], is defined by the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand}}(\mathfrak{K})$ depicted in Figure 6. It is a strong version of the pseudo-randomness where the adversary is also given the hash value of a commitment of its choice (obviously not generated by SCom or SimCom though, hence the test with Λ which also contains (C, ℓ, x)). This property only makes sense when the projection key does not depend on the word C to be hashed. It thus applies to $\mathsf{KV\text{-}SPHF}$, and $\mathsf{CS\text{-}SPHF}$ only.

B Groth Sahai Methodology

Groth and Sahai [GS08] have introduced a methodology to build Non-Interactive ZK / Witness Indistinguishable proofs of *satisfiability of pairing-product like equations.* The three types of equations handled by such proofs are the following:

A *pairing-product equation* over variables $\boldsymbol{\mathcal{X}} = (\mathcal{X}_1, \ldots, \mathcal{X}_m) \in \mathbb{G}_1^m$ and $\boldsymbol{\mathcal{Y}} = (\mathcal{Y}_1, \ldots, \mathcal{Y}_n) \in \mathbb{G}_2^n$ is of the form

$$\langle \boldsymbol{\mathcal{A}}, \underline{\boldsymbol{\mathcal{Y}}} \rangle \cdot \langle \underline{\boldsymbol{\mathcal{X}}}, \boldsymbol{\mathcal{B}} \rangle \cdot \langle \underline{\boldsymbol{\mathcal{X}}}, \Gamma \underline{\boldsymbol{\mathcal{Y}}} \rangle = t_T \ , \tag{1}$$

defined by constants $\boldsymbol{\mathcal{A}} \in \mathbb{G}_1^n$, $\boldsymbol{\mathcal{B}} \in \mathbb{G}_2^m$, $\Gamma = (\gamma_{i,j})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} \in \mathbb{Z}_p^{m \times n}$ and $t_T \in \mathbb{G}_T$.

A *multi-scalar multiplication equation* over variables $\boldsymbol{y} \in \mathbb{Z}_p^n$ and $\boldsymbol{\mathcal{X}} \in \mathbb{G}_1^m$ is of the form

$$\langle \underline{\boldsymbol{y}}, \boldsymbol{\mathcal{A}} \rangle \cdot \langle \boldsymbol{b}, \underline{\boldsymbol{\mathcal{X}}} \rangle \cdot \langle \underline{\boldsymbol{y}}, \Gamma \underline{\boldsymbol{\mathcal{X}}} \rangle = T, \tag{2}$$

defined by the constants $\boldsymbol{\mathcal{A}} \in \mathbb{G}_1^n$, $\boldsymbol{b} \in \mathbb{Z}_p^m$, $\Gamma \in \mathbb{Z}_p^{m \times n}$ and $T \in \mathbb{G}_1$.

A multi-scalar multiplication equation in group $\mathbb{G}_2$ is defined analogously.

A *quadratic equation in* $\mathbb{Z}_p$ over variables $\boldsymbol{x} \in \mathbb{Z}_p^m$ and $\boldsymbol{y} \in \mathbb{Z}_p^n$ is of the form

$$\langle \boldsymbol{a}, \underline{\boldsymbol{y}} \rangle + \langle \underline{\boldsymbol{x}}, \boldsymbol{b} \rangle + \langle \underline{\boldsymbol{x}}, \Gamma \underline{\boldsymbol{y}} \rangle = t, \tag{3}$$

defined by the constants $\boldsymbol{a} \in \mathbb{Z}_p^n$, $\boldsymbol{b} \in \mathbb{Z}_p^m$, $\Gamma \in \mathbb{Z}_p^{m \times n}$ and $t \in \mathbb{Z}_p$.

Groth and Sahai have detailed generic construction of the proofs π and specific instantiations under different security assumptions. We will focus on Linear Equations in the following, as they are those needed in the rest of the paper, that is to say equations with variables in only one of the two groups.

B.1 SXDH Instantiation

In order to generate a proof of such relations, the methodology invites us to commit to the witness vectors $\boldsymbol{\mathcal{X}}$ with randomness $\boldsymbol{R}$, and to $\boldsymbol{\mathcal{Y}}$ with $\boldsymbol{S}$ with two double ElGamal commitments scheme, one in $\mathbb{G}_1$ and one in $\mathbb{G}_2$ with respective commitment keys $\mathbf{u} \in \mathbb{G}_1^{2\times 2}$ and $\mathbf{v} \in \mathbb{G}_2^{2\times 2}$. As both need to be semantically secure, we will work under the SXDH assumption.

We will note $\iota_1(g_1)=(1_1, g_1)$, $\iota_2(g_2)=(1_2, g_2)$, $\iota_T(t_T) := \begin{pmatrix} 1_T & 1_T \\ 1_T & t_T \end{pmatrix}$ and focus on product pairing equations.

Assuming elements were committed in $\mathbb{G}_2$ following the way explained in 2, following [GS08] notations, we then have access to algorithms:

– $\mathsf{Prove}((\mathcal{Y}_j), (\mathbf{u}, \mathbf{v}), E; (S_j), T \in \mathbb{Z}_p^{2\times 2})$ outputs a proof π, together with $\boldsymbol{d} \in \mathbb{G}_2^{2\times n}$ commitments to the witnesses with randomness $\boldsymbol{S} \in \mathbb{Z}^{2\times n}$. The proof is composed of two elements in $\mathbb{G}_1$: $\pi = \boldsymbol{S}^\top \boldsymbol{\mathcal{A}}$

– $\mathsf{Verify}(\boldsymbol{d}, \mathsf{ck}, E, \pi)$ checks if: $\big(\iota_1(\boldsymbol{\mathcal{A}}) \bullet \boldsymbol{d}\big) = \iota_T(t_T) \odot \big(\iota_1(\pi) \bullet \mathbf{v}\big)$

There is also an algorithm that allows anyone to randomize the proof, even without the knowledge of the witnesses.

The *Soundness* and the *Witness Indistinguishability* of such a proof directly come from the security of the commitment, and the extra randomness T.

Intuitively the proof is here to compensate some part introduced by the randoms in the verification equation: $\boldsymbol{S}^\top \boldsymbol{\mathcal{A}}$ will cancel the randomness in $\big(\iota_1(\boldsymbol{\mathcal{A}}) \bullet \boldsymbol{d}\big)$.

B.2 k-MDDH Instantiation

[EHK+13] defined a generalization of the Groth and Sahai framework for Zero-Knowledge proof to fit with the Matrix Assumption.

Given an underlying matrix $[\boldsymbol{B}]_s$, one generate a binding CRS, by computing an additional vector in the span of $[\boldsymbol{B}]_s$:$[\boldsymbol{Bw}]_s$, and a hiding one, if there is some perturbation to this column. We name the commitment matrix $[\boldsymbol{U}]$ which is the concatenation of the matrix $\boldsymbol{B}$ and this extra column.

We define the $k+1$ vector: $\iota_s(\mathcal{X}) := (1, \ldots, 1, \mathcal{X})$, and the $k+1$ by $k+1$ matrix $\iota_T(t_T) := \begin{pmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & 1 \\ 1 & 1 & t_T \end{pmatrix}$. This allows us to extend the previous commitment to their equivalent under $k-$MDDH.

To commit to an element M, one computes $[\boldsymbol{R}]_s$ with randomness $\rho \xleftarrow{\$} \mathbb{Z}_p^{k+1}$, setting $[\boldsymbol{R}]_2 = [\boldsymbol{U}\boldsymbol{\rho} + \iota_s(M)]_s \in \mathbb{G}_2^{k+1}$.

As before, there exist $\mathsf{Prove}, \mathsf{Verify}$ algorithms allowing to generate a compatible, randomizable, Zero-Knowledge proof that the committed value fulfills some pairing equation.

C Ideal Functionalities

C.1 UC Framework

The goal of the UC framework is to ensure that UC-secure protocols will continue to behave in the ideal way even if executed in a concurrent way in arbitrary environments. It is a simulation-based model, relying on the indistinguishability between the real world and the ideal world. In the ideal world, the security is provided by an ideal functionality $\mathcal{F}$, capturing all the properties required for the protocol and all the means of the adversary. In order to prove that a protocol Π emulates $\mathcal{F}$, one has to construct, for any polynomial adversary $\mathcal{A}$ (which controls the communication between the players), a simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ (the distinguisher) can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage. The adversary can be either *adaptive*, *i.e.* allowed to corrupt users whenever it likes to, or *static*, *i.e.* required to choose which users to corrupt prior to the execution of the session sid of the protocol. After corrupting a player, $\mathcal{A}$ has complete access to the internal state and private values of the player, takes its entire control, and plays on its behalf.

C.2 UC-Secure Oblivious Transfer

The ideal functionality of an Oblivious Transfer (OT) protocol is depicted in Figure 7. It is inspired from [CKWZ13].

The functionality $\mathcal{F}_{(1,k)\text{-OT}}$ is parameterized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1,\ldots,P_n$ via the following queries:

- **Upon receiving an input $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ from party P_i**, with $m_i \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ from party P_j**, with $s \in \{1, \ldots, k\}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$, and reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$. Ignore further Receive-message with the same ssid from P_j.
- **Upon receiving a message $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Sent}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to P_i and ignore further Sent-message with the same ssid from the adversary.
- **Upon receiving a message $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from the adversary $\mathcal{S}$**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, (m_1, \ldots, m_k))$ or $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, s)$ is not recorded; otherwise send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, m_s)$ to P_j and ignore further Received-message with the same ssid from the adversary.

Fig. 7. Ideal Functionality for 1-out-of-k Oblivious Transfer $\mathcal{F}_{(1,k)\text{-OT}}$

C.3 UC-Secure Password-Authenticated Key Exchange

We present the PAKE ideal functionality $\mathcal{F}_{pwKE}$ on Figure 8. It was described in [CHK+05].

The main idea behind this functionality is as follows: If neither party is corrupted and the adversary does not attempt any password guess, then the two players both end up with either the same uniformly-distributed session key if the passwords are the same, or uniformly-distributed independent session keys if the passwords are distinct. In addition, the adversary does not know whether this is a success or not. However, if one party is corrupted, or if the adversary successfully guessed the player's password (the session is then marked as `compromised`), the adversary is granted the right to fully determine its session key. There is in fact nothing lost by allowing it to determine the key. In case of wrong guess (the session is then marked as `interrupted`), the

The functionality $\mathcal{F}_{\text{pwKE}}$ is parameterized by a security parameter k. It interacts with an adversary $\mathcal{S}$ and a set of parties $P_1, \ldots, P_n$ via the following queries:

- **Upon receiving a query $(\texttt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ from party P_i:**
  Send $(\texttt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to $\mathcal{S}$. If this is the first `NewSession` query, or if this is the second `NewSession` query and there is a record $(\mathsf{sid}, \mathsf{ssid}, P_j, P_i, \mathsf{pw}')$, then record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ and mark this record `fresh`.
- **Upon receiving a query $(\texttt{TestPwd}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{pw}')$ from the adversary S:**
  If there is a record of the form (P_i, P_j, pw) which is `fresh`, then do: If $pw = pw'$, mark the record `compromised` and reply to $\mathcal{S}$ with "correct guess". If $\mathsf{pw} \neq \mathsf{pw}'$, mark the record `interrupted` and reply with "wrong guess".
- **Upon receiving a query $(\texttt{NewKey}, \mathsf{sid}, \mathsf{ssid}, P_i, \mathsf{sk})$ from the adversary S:**
  If there is a record of the form $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$, and this is the first `NewKey` query for P_i, then:
  - If this record is `compromised`, or either P_i or P_j is corrupted, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk})$ to player P_i.
  - If this record is `fresh`, and there is a record (P_j, P_i, pw') with $\mathsf{pw}' = \mathsf{pw}$, and a key sk' was sent to P_j, and (P_j, P_i, pw) was `fresh` at the time, then output $(\mathsf{sid}, \mathsf{ssid}, \mathsf{sk}')$ to P_i.
  - In any other case, pick a new random key sk' of length $\mathfrak{K}$ and send $(\mathsf{sid}, \mathsf{ssid}, sk')$ to P_i.

  Either way, mark the record $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{pw})$ as `completed`.

Fig. 8. Ideal Functionality for PAKE $\mathcal{F}_{\text{pwKE}}$

two players are given independently-chosen random keys. A session that is nor `compromised` nor `interrupted` is called `fresh`, which is its initial status.

Finally notice that the functionality is not in charge of providing the password(s) to the participants. The passwords are chosen by the environment which then hands them to the parties as inputs. This guarantees security even in the case where two honest players execute the protocol with two different passwords: This models, for instance, the case where a user mistypes its password. It also implies that the security is preserved for all password distributions (not necessarily the uniform one) and in all situations where the password, are related passwords, are used in different protocols. Also note that allowing the environment to choose the passwords guarantees forward secrecy.

In case of corruption, the adversary learns the password of the corrupted player, after the `NewKey`-query, it additionally learns the session key.

D Proof of the Strong Pseudo-Randomness

Proof. To prove the strong pseudo-randomness, we use the following sequence of games:

Game G_0: This game is the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-0}}$.

Game G_1: In this game, we replace $(\boldsymbol{C}, \boldsymbol{R}, \mathsf{eqk}) \xleftarrow{\$} \mathsf{SimCom}^{\ell}(\tau)$ by $\boldsymbol{C}, \boldsymbol{R}, \xleftarrow{\$} \mathsf{Com}^{\ell}(\boldsymbol{M''})$ for some arbitrary $\boldsymbol{M''} \neq \boldsymbol{M}$. This game is indistinguishable thanks to strong simulation indistinguishability (Commitments and fake commitments are the same, only their decommitment witnesses differs but they are indistinguishably distributed).

Game G_2: In this game, we replace the CRS, by a perfectly binding one. (There are no more equivocation used, only real commitments). Under k-MDDH, this is indistinguishable from the previous one.

Game G_3: In this game, before computing H', we compute $\boldsymbol{M'} \leftarrow \mathsf{ExtCom}^{\ell'}(\tau, C')$ and we abort if the decryption is not unique. In other words, if C' is not perfectly binding, we abort. However as we now have a perfectly hiding CRS, this never happen. This game is indistinguishable from the previous one.

Game G_4: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H' by a random value.

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, the fact that $\boldsymbol{M'} \neq \boldsymbol{M}$ and C' is perfectly binding (otherwise, we would have aborted), so that $(\ell', \boldsymbol{C'}, \boldsymbol{R'}) \notin L_{\boldsymbol{M}}$, and thanks to the fact that H could have been computed as follows: $\Pi \leftarrow \mathsf{OpenCom}^{\ell}(\mathsf{eqk}, \boldsymbol{C}, \boldsymbol{R}, \boldsymbol{M})$ and $H \leftarrow \mathsf{ProjHash}(\mathsf{hp}, L_{\boldsymbol{M}}, (\ell, \boldsymbol{C}, \boldsymbol{R}), \Pi)$.

Game G_5: In this game, when $\boldsymbol{M'} \neq \boldsymbol{M}$, we replace H by a random value.

This game is indistinguishable from the previous one thanks to the smoothness of the SPHF, and the fact that C is a real commitment of $\boldsymbol{M''} \neq \boldsymbol{M}$ and so that $(\ell, \boldsymbol{C}, \boldsymbol{R}) \notin L_{\boldsymbol{M}}$.
Notice that we could not have done this if $\boldsymbol{M'} = \boldsymbol{M}$, since, in this case, we still need to use hk to compute the hash value H' of C'. We are handling this (tricky) case in the following game.

Game $\boldsymbol{G}_6$: In this game, we replace H by a random value, in the case $\boldsymbol{M'} = \boldsymbol{M}$. So now H will be completely random, in all cases (since it was already the case when $\boldsymbol{M'} \neq \boldsymbol{M}$).
Finally, we write $[\boldsymbol{r'}]_2$, the vector extracted from $\boldsymbol{R}'$. There are two cases:

1. If $[\boldsymbol{C}'_1]_T = [\boldsymbol{A} \cdot \boldsymbol{r}']_T$ In this case, since C' extracts to $\boldsymbol{M}$, this means that $(\ell', \boldsymbol{C}', \boldsymbol{R}') \in L_{\boldsymbol{M}}$, and its hash value H' could be computed knowing only hp and r'. Therefore, the hash value H of C looks random by smoothness.
2. Else $\boldsymbol{C}'_1$ is not in the correct span, the rows of the matrix Γ in Equation and the two vectors $\Theta(\boldsymbol{C})$ and $\Theta(\boldsymbol{C}')$ are linearly independent. Then, even given access to the hash value H' of $\boldsymbol{C}'$ and the projection key hp, the hash value H of $\boldsymbol{C}$ looks perfectly random.

The following games are just undoing the modifications we have done, but keeping H picked at random

Game $\boldsymbol{G}_7$: In this game, if $\boldsymbol{M'} \neq \boldsymbol{M}$, we compute H' as originally (as the hash value of C').
This game is indistinguishable from the previous one thanks to the smoothness of the SPHF.

Game $\boldsymbol{G}_8$: In this game, we do not extract $\boldsymbol{M'}$ from C' nor abort when C' is not perfectly binding.

Game $\boldsymbol{G}_9$: In this game, we now move back to the perfectly hiding CRS. Under k-MDDH this game is indistinguishable from the previous one.

Game $\boldsymbol{G}_{10}$: In this game, we now compute C as originally using SimCom. This game is indistinguishable thanks to strong simulation indistinguishability.
We remark that this game is exactly the experiment $\mathsf{Exp}_{\mathcal{A}}^{\texttt{c-s-ps-rand-1}}$.

□

E Proof of the Oblivious Transfer Scheme

To prove theorem 11, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}$ and the adversary $\mathcal{A}$.

Compared to the protocol presented in [ABB$^+$13], the difficulty here arises from the fact that the commitment from [FLM11] needs the CRS to be hiding in order to be equivocable and that such a CRS forbids the use of smoothness for the SPHF. Instead of being able to replace all the commitment queries by simulated (fake) commitments from the beginning of the proof, one has to only allow the extraction trapdoors (for the Cramer-Shoup encryption inside the commitment, and for the CPA encryption of J) and to deal carefully with the properties of the SPHF before being able, at the end of the proof, to turn the CRS into a hiding one and simulate the commitments. More details follow (the description of the simulator can be found in the last game).

Game $\boldsymbol{G}_0$: This is the real game.

Game $\boldsymbol{G}_1$: In this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs $(n_1, \dots, n_t)$ and j sent by the environment to the server and the user. In all the subsequent games, the players use the label $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

Game $\boldsymbol{G}_2$: In this game, we just replace the setup algorithms so that the simulator knows the trapdoor for extracting both the Cramer-Shoup encryption, and the $\mathsf{Encrypt}_{\mathrm{cpa}}$ encryption. Note that we do not change anything more in the setup, implying the CRS remains a CRS for a perfectly-sound Groth-Sahai setting. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: We first deal with **honest servers**: when receiving a commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$, the simulator extracts the committed value j from $[\boldsymbol{C}]_1$. Instead of computing the key K_s, for $s = 1, \dots, t$ with the hash function, it chooses $K_s \xleftarrow{\$} G$ for $s \neq j$.
Since $[\boldsymbol{C}]_1$ is extracted to j, then, with an hybrid proof applying the smoothness for every honest server, on every index $s \neq j$, the hash value K_s is indistinguishable from a random value for $s \neq j$.
In case of corruption, everything has been erased. This game is thus indistinguishable from the previous one under the smoothness of the smooth projective hash function.

Game $\boldsymbol{G}_4$: We continue to deal with **honest servers**: when receiving a commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$, the simulator extracts the committed value j from $[\boldsymbol{C}]_1$. Instead of proceeding as the server would do on $(n_1, \dots, n_t)$, the simulator proceeds on $(n'_1, \dots, n'_t)$, with $n'_j = n_j$, but $n'_s = 0$ for all $s \neq j$. Since the masks K_t, for $t \neq s$, are random from the previous game, this game is perfectly indistinguishable from the previous one.

Game $\boldsymbol{G}_5$: We continue to deal with **honest servers** and more precisely with the computation of S. If the user and the server are still honest until the server has received an honestly-generated $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, c)$ from the honest user, the simulator does not extract J and compute $S \leftarrow F(J)$, but directly sets $S \leftarrow (J')$, with J' a random value, for both players.
With an hybrid proof, applying the $\mathsf{IND\text{-}CPA}$ property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_6$: We continue to deal with **honest servers** and more precisely with the case described in the previous game. In this case, the simulator now directly sets S as a random value, instead of setting $S \leftarrow F(J')$.
With an hybrid proof, applying the PRF property for each session, one can show the indistinguishability of this game with the previous one.

Game $\boldsymbol{G}_7$: We continue to deal with **honest servers** and now describe how the simulator generates K_j, in case the user and the server are still honest until the server has received an honestly-generated $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, c)$ from the honest user. In this case, thanks to the additional random mask S, the simulator can send a random N_j on behalf of the server, and postpone the computation of K_j at the time the user actually receives this value.
Since the adversary does not know any decommitment information $[\boldsymbol{\Pi}]_1$ for the commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$ (thanks to the $\mathsf{IND\text{-}CCA}$ properties of the Cramer-Shoup and Groth-Sahai encryptions), this hash value K_j is indistinguishable from a random value, applying the pseudo-randomness for every honest server. If the server involved in the pseudo-randomness gets corrupted, we are out of this case and can thus abort it.
In case of corruption of the server, everything has been erased. In case of corruption of the user, the simulator receives the good value n_j and is able to choose R (which is a random value unknown to the adversary, and because all the other K_t are independent random values too) such that

$$R \oplus \mathsf{ProjHash}(\mathsf{hp}_j, (L_j, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2)) \oplus N_j = n_j.$$

This game is thus indistinguishable from the previous one under the pseudo-randomness.

Game $\boldsymbol{G}_8$: We continue to deal with the same case. On behalf of an honest server, the simulator proceeds with the database $(n'_1, \dots, n'_s)$, with $n'_s = 0$ for all s even $s = j$. Since the masks $K_s \oplus S$, for any $s = 1, \dots, t$, are all independent random values (this comes from the fact that the K_t, for $t \neq s$, are independent random values, and S is independently random), this game is perfectly indistinguishable from the previous one.
We remark that it is therefore no more necessary to know the index j given by the ideal functionality to the honest user in order to correctly simulate the server. But note that

the knowledge of this index is still necessary to simulate the user (in particular in case of corruption). We show in the next games how to get rid of this knowledge.

Game $\boldsymbol{G}_9$: In this game, we completely replace the setup algorithms so that not only the simulator knows the trapdoor for extracting both the Cramer-Shoup encryption, and the $\mathsf{Encrypt}_{\mathrm{cpa}}$ encryption, but the CRS also becomes a CRS for a witness indistinguishable Groth-Sahai setting. This game is indistinguishable from the former one under the k-MDDH assumption.

Game $\boldsymbol{G}_{10}$: In this game, we deal with **honest users**, assuming that the simulator still knows the index j which has to be committed to by the honest users and still computes the commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2)$ honestly. However, in case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ using the simulation trapdoor rather than the real witnesses. This game is indistinguishable from the former one since, in the witness-indistinguishable setting, simulated proofs are distributed as real proofs.

Game $\boldsymbol{G}_{11}$: In this game, we still deal with **honest users**, by without using anymore the knowledge of j when simulating. On behalf of an honest user, the simulator chooses an index j' at random, and computes honestly the commitments $([\boldsymbol{C}]_1, [R_2])$ to this value j'. In case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ corresponding to the real j it has just learnt using the simulation trapdoor as in the former game. Since the proofs are already simulated, this is indistinguishable from the former game thanks to the semantic security of the Cramer-Shoup encryption. The argument uses an hybrid proof and is the same as [CF01, Theorem 8] or [FLM11, Theorem 1].
When it finally receives the values (hp_s, N_s) from the adversarial server, the simulator computes, for $s = 1, \ldots, t$, $[\boldsymbol{\Pi}_s]_1$ corresponding to a commitment of s, $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, (L_s, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}_s]_1)$ and gets $n_s \leftarrow S \oplus K_s \oplus N_s$, giving it the database submitted by the server. The only problem would arise if an adversarial user committed to j and was able to open its commitment to j' by computing the correct $[\boldsymbol{\Pi}_{j'}]_1$. But since the commitment is universally composable, there is a neglibible probability for it to be able to compute this witness. Thus, the security again relies on the pseudo-randomness, making this game indistinguishable from the previous one.

Game $\boldsymbol{G}_{12}$: We can now make use of the functionality, which leads to the following simulator:
- when receiving a Send-message from the ideal functionality, which means that an honest server has sent a pre-flow, the simulator generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as pre-flow;
- after receiving a pre-flow pk (from an honest or a corrupted server) and a `Receive`-message from the ideal functionality, which means that an honest user has sent an index query, the simulator generates $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \overset{\$}{\leftarrow} \mathsf{Com}(\mathsf{crs}, j', \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ for a random index j' and $c \overset{\$}{\leftarrow} \mathsf{Encrypt}(\mathsf{pk}, S)$, for a random value S, and sends $[\boldsymbol{C}]_1$, $[\boldsymbol{R}]_2$ and c during the index query phase on behalf of the honest user;
- when receiving a commitment $[\boldsymbol{C}]_1$, $[\boldsymbol{R}]_2$ and a ciphertext c, generated by the adversary (from a corrupted user), the simulator extracts the committed value j, and uses it to send a `Receive`-message to the ideal functionality. It also decrypts the ciphertext c as J, and computes $S = F(J)$;
- when receiving $(\mathsf{hp}_1, N_1, \ldots, \mathsf{hp}_t, N_t)$ from the adversary (a corrupted server), the simulator computes, for $i = 1, \ldots, t$, $[\boldsymbol{\Pi}_s]_1$ corresponding to a commitment of s, $K_s \leftarrow \mathsf{ProjHash}(\mathsf{hp}_s, (L_s, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}_s]_1)$ and gets $n_s \leftarrow S \oplus K_s \oplus N_s$, giving it the database submitted by the server. It uses these values to send a Send-message to the ideal functionality.
- when receiving a `Received`-message from the ideal functionality, together with n_j, on behalf of a corrupted user, from the extracted j, instead of proceeding as the server would do on $(n_1, \ldots, n_t)$, the simulator proceeds on $(n'_1, \ldots, n'_t)$, with $n'_j = n_j$, but $n'_s = 0$ for all $s \neq t$;

– when receiving a commitment $[\boldsymbol{C}]_1$, $[\boldsymbol{R}]_2$ and a ciphertext c, generated by an honest user (*i.e.*, by the simulator itself), the simulator proceeds as above on $(n'_1, \dots, n'_t)$, with $n'_s = 0$ for all s, but it chooses S uniformly at random instead of choosing it as $S = F(J)$; in case of corruption afterwards, the simulator will adapt S such that $S \oplus \mathsf{ProjHash}(\mathsf{hp}_j, \ell, (L_j, [C]_1, [R]_2), [\varPi]_1) \oplus N_j = n_j$, where n_j is the message actually received by the user.

Any corruption either reveals j earlier, which allows a correct simulation of the user, or reveals $(n_1, \dots, n_t)$ earlier, which allows a correct simulation of the server. When the server has sent his flow, he has already erased all his random coins.

However, there would have been an issue when the user is corrupted after the server has sent is flow, but before the user receives it, since he has kept $\varPi_1$: this would enable the adversary to recover n_j from N_j and hp_j. This is the goal of the ephemeral mask S that provides a secure channel.

F Proof of the PAKE Scheme

To prove theorem 12, we exhibit a sequence of games. The sequence starts from the real game, where the adversary $\mathcal{A}$ interacts with real players and ends with the ideal game, where we have built a simulator $\mathcal{S}$ that makes the interface between the ideal functionality $\mathcal{F}_{\text{PAKE}}$ and the adversary $\mathcal{A}$. For simplicity, in the proof we are going to use pw loosely to designate an encoding $G(\mathsf{pw})$. A decryption can lead to two cases, either a valid encoding $G(\mathsf{pw})$ which can then be reverted to a password pw, or an invalid one. In this last case, the simulator assumes the password pw to be $\perp$.

For the sake of simplicity, since the protocol is fully symmetric in P_i and P_j, we describe the simulation for player P_i in order to simplify the notations.

We say that a flow is *oracle-generated* if the tuple $(\mathsf{hp}_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2)$ was sent by an honest player P_i (or the simulator) and received without any alteration by the adversary. It is said *non-oracle-generated* otherwise.

Game $\boldsymbol{G}_0$: This is the **real game.**

Game $\boldsymbol{G}_1$: First, in this game, the simulator generates correctly every flow from the honest players, as they would do themselves, knowing the inputs pw_i and pw_j sent by the environment to the players. In case of corruption, the simulator can give the internal data generated on behalf of the honest players.

In the following, Step 1. is always generated honestly by the simulator, since the hashing and projection keys do not depend on any private value.

Game $\boldsymbol{G}_2$: In this game, we just replace the setup algorithms so that the simulator knows the trapdoor for extracting the Cramer-Shoup encryption. Note that we do not change anything more in the setup, implying the CRS remains a CRS for a perfectly-sound Groth-Sahai setting. Corruptions are handled the same way.

Game $\boldsymbol{G}_3$: In this game, we deal with the case where P_i **receives a flow oracle-generated from P_j, and they have identical passwords.** When P_i receives an oracle-generated flow from P_j, the simulator checks whether the two passwords sent by the environment for P_i and P_j are identical. If so, $\mathcal{S}$ computes both hash values using Hash and not $\mathsf{ProjHash}$. More precisely, it computes $H'_i = \mathsf{Hash}(\mathsf{hk}_j, (\mathcal{L}_{\mathsf{pw}_j}, \ell_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2)))$ (with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$). If the passwords are distinct, it does not change anything. Recall that it is able to do so since it generated the hashing keys on their behalf.

Thanks to the correctness of the SPHF, this game is indistinguishable from the former one.

Game $\boldsymbol{G}_4$: In the next two games, we deal with the case where P_i **receives a flow oracle-generated from P_j, but P_j has been corrupted, and they have distinct passwords**.

In this case, $\mathcal{S}$ has received the password pw_j of P_j at the corruption time of P_j (pw_j was anyway already known), and knows the corresponding opening data $[\mathit{\Pi}_j]_1$, which it computed honestly on behalf of P_j (since it still knows pw_j). If this password is the same, it does not change anything. If the passwords are distinct, then $\mathcal{S}$ computes H'_i as before, but chooses H_j at random: this means that we replace $\mathsf{Hash}(\mathsf{hk}_i, (\mathcal{L}_{\mathsf{pw}_i}, \ell_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2))$ by a random value, while $[\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2$ have been simulated by Com with an opening value $[\mathit{\Pi}_j]_1$ for $\mathsf{pw}_j \neq \mathsf{pw}_i$. Using an hybrid proof, this game is indistinguishable from the former one using the smoothness of the SPHF.

Game $\boldsymbol{G}_5$: We conclude for this case: if the passwords are distinct, P_i chooses a random key. Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_6$: In the next two games, we deal with the case where P_i **receives a flow oracle-generated from P_j, and P_j is still honest, and they have distinct passwords.** The simulator checks whether the two passwords sent by the environment for P_i and P_j are distinct. If so, $\mathcal{S}$ replaces $\mathsf{Hash}(\mathsf{hk}_j, (\mathcal{L}_{\mathsf{pw}_j}, \ell_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2))$ by a random value, the first time it is computed (and uses the same random value the second time it is computed by the partner, and thus only if this is the same password).

In case the player on which we made the modification is later corrupted, this is out of this case, and thus we abort this hybrid game and go to the next one. Using an hybrid proof, this game is indistinguishable from the former one using the pseudo-randomness.

Game $\boldsymbol{G}_7$: We conclude for this case: $\mathcal{S}$ sends a random key to P_i. Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_8$: In the next two games, we deal with the case where P_i **receives a non-oracle-generated flow** $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$. Since this pair is fresh, either $[\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2$ is new or hp_j (and thus the label) is new. Since $[\boldsymbol{R}_j]_2$ is used in the label of the Cramer-Shoup encryption $[\boldsymbol{C}_j]_1$, either both of them are new or not. In both cases, $\mathcal{S}$ can extract the committed value pw'_j on behalf of P_j.

If this password is the same than that of P_i (which the simulator can easily check, still having access to the private values sent by the environment), $\mathcal{S}$ still computes both H_j and H'_i as before.

Otherwise (or if the extraction fails), the $\mathcal{S}$ computes H'_i as before, but chooses H_j at random: Under the smoothness, with an hybrid proof, one can show the indistinguishability of the two games.

Game $\boldsymbol{G}_9$: Finally, when P_i receives a non-oracle-generated flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ that extracts to a different password than that of P_i (or for which extraction fails), then $\mathcal{S}$ sets the session key of P_i as random.

Since this is a simple syntactical change from the former game, this game is perfectly indistinguishable from it.

Game $\boldsymbol{G}_{10}$: In this game, we completely replace the setup algorithms so that not only the simulator knows the trapdoor for extracting the Cramer-Shoup encryption, but the CRS also becomes a CRS for a witness indistinguishable Groth-Sahai setting. This game is indistinguishable from the former one under the k-MDDH assumption.

Game $\boldsymbol{G}_{11}$: In this game, we still use the knowledge of pw_i to compute $[\boldsymbol{C}]_1, [\boldsymbol{R}]_2$ but in case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ using the simulation trapdoor rather than the real witnesses. This game is indistinguishable from the former one since, in the witness-indistinguishable setting, simulated proofs are distributed as real proofs.

Game $\boldsymbol{G}_{12}$: We do not use anymore the knowledge of pw_i when simulating an honest player P_i. On behalf of an honest user, the simulator chooses a password pw'_i at random, and computes honestly the commitments $([\boldsymbol{C}]_1, [R_2])$ to this value pw'_i. In case of corruption, it computes the proof $[\boldsymbol{\Pi}]_1$ corresponding to the real pw_i it has just learnt using the simulation trapdoor as in the former game. Since the proofs are already simulated, this is indistinguishable from the former game thanks to the semantic security of the Cramer-Shoup encryption. The argument uses an hybrid proof and is the same as [CF01, Theorem 8] or [FLM11, Theorem 1].

The only problem would arise if an adversarial user committed to pw'_i and was able to open its commitment to pw''_i by computing the correct $[\boldsymbol{\Pi}]_1$. But since the commitment is universally composable, there is a negligible probability for it to be able to compute this witness. Thus, the security again relies on the pseudo-randomness, making this game indistinguishable from the previous one.

The private values of P_i are thus not used anymore in Step 1. and Step 2. The simulator only needs them to choose how to set the session key of the players. In the ideal game, this will be replaced by a NewKey-query that will automatically deal with equality or difference of the passwords, or $\mathtt{TestPwd}$-query for non-oracle-generated-flows.

Game $\boldsymbol{G}_{13}$: This is the ideal game. Now, the simulator does not know the private values of the honest players anymore, but can make use of the ideal functionality. We showed in Game $\boldsymbol{G}_{12}$ that the knowledge of the private values is not needed anymore by the simulator, provided it can ask queries to the ideal functionality:

Initialization: When initialized with security parameter $\mathfrak{K}$, the simulator first runs the commitment setup algorithm obtaining a witness indistinguishable crs in which it knows the trapdoors for the extraction of the Cramer-Shoup encryption and the simulation of the Groth-Sahai proofs. It initializes the real-world adversary $\mathcal{A}$, giving it crs as common reference string.

Session Initialization: When receiving a message $(\mathtt{NewSession}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ from $\mathcal{F}_{pwKE}$, $\mathcal{S}$ executes the protocol on behalf of P_i as follows:

1. $\mathcal{S}$ generates honestly $\mathsf{hk}_i \overset{\$}{\leftarrow} \mathsf{HashKG}(\mathcal{L})$ and $\mathsf{hp}_i \leftarrow \mathsf{ProjKG}(\mathsf{hk}_i, \mathcal{L})$;
2. $\mathcal{S}$ computes $([\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2, [\boldsymbol{\Pi}_i]_1)\mathsf{Com}^{\ell_i}(\mathsf{crs}, \mathsf{pw}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell_i = (\mathsf{sid}, P_i, P_j, \mathsf{hp}_i)$;
3. $\mathcal{S}$ sends $\mathsf{hp}_i, [\boldsymbol{C}_i]_1, [\boldsymbol{R}_i]_2$ to P_j.

If P_i gets corrupted, $\mathcal{S}$ recovers the password pw_i and computes the corresponding proof $[\Pi_i]_1$, which it is able to give to the adversary.

Key Computation: When receiving a flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$:

- if the flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ is non-oracle-generated, $\mathcal{S}$ extracts the password pw'_j (or sets it as a dummy value in case of failure of extraction). $\mathcal{S}$ then asks for a $\mathtt{TestPwd}$-query to the functionality to check whether pw'_j is the password of P_i. If this password is correct, $\mathcal{S}$ sets $\mathsf{pw}_i = \mathsf{pw}'_j$, computes the corresponding proof $[\Pi_i]_1$, as well as H_j and H'_i, and then sk_i, that is passed to the NewKey-query ($\mathtt{compromised}$ case). If the password is incorrect, $\mathcal{S}$ asks the NewKey-query with a random key ($\mathtt{interrupted}$ case).
- if the flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ is oracle-generated but the associated P_j has been corrupted, then $\mathcal{S}$ has recovered its password pw_j and has been able to compute the corresponding proof $[\Pi_j]_1$. It can thus compute sk_j, that is passed to the NewKey-query ($\mathtt{corrupted}$ case).
- if the flow $(\mathsf{hp}_j, [\boldsymbol{C}_j]_1, [\boldsymbol{R}_j]_2)$ is oracle-generated and the associated P_j is still uncorrupted, $\mathcal{S}$ asks the NewKey-query with a random key (normal case).

One can remark that the NewKey-queries will send back the same kinds of session keys to the environment as in Game $\boldsymbol{G}_{12}$: if a player is corrupted, the really computed key is sent back, in case of impersonation attempt, the $\mathtt{TestPwd}$-query will address the appropriate situation (correct or incorrect guess), and if the two players are honest, the NewKey-query also addresses the appropriate situation (same or different passwords).

G Sketch of Proof for Anonymous Credential-Based Message Transmission

G.1 Ideal Functionality

The ideal functionality for Anonymous Credential-Based Message Transmission is given in Figure 9. The server S agrees to send a message M to the user, as soon as his credentials Cred comply with the policy P.

The functionality $\mathcal{F}_{\mathrm{AC}}$ is parametrized by a security parameter $\mathfrak{K}$. It interacts with an adversary $\mathcal{S}$ and a set of parties P_1,...,P_N via the following queries:

- **Upon receiving an input $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, M, P)$ from party P_i**, with $M \in \{0,1\}^{\mathfrak{K}}$: record the tuple $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, M, P)$ and reveal $(\mathsf{Send}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, P)$ to the adversary $\mathcal{S}$. Ignore further Send-message with the same ssid from P_i.
- **Upon receiving an input $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, \mathsf{Cred})$ from party P_j**: ignore the message if $(\mathsf{sid}, \mathsf{ssid}, P_i, P_j, M, P)$ is not recorded. Otherwise, reveal $(\mathtt{Receive}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ to the adversary $\mathcal{S}$ and send $(\mathtt{Received}, \mathsf{sid}, \mathsf{ssid}, P_i, P_j, M')$ to P_j where $M' = M$ if the credentials comply with the policy P, and $M' = \perp$ otherwise. Ignore further $\mathtt{Receive}$-message with the same ssid from P_j.

Fig. 9. Ideal Functionality for Anonymous Credential-Based Message Transmission $\mathcal{F}_{\mathrm{AC}}$

G.2 Idea of the Proof

We sketch the proof by giving the simulator $\mathcal{S}$ such that no polynomial environment $\mathcal{Z}$ can distinguish between the real world (with the real players interacting with themselves and $\mathcal{A}$ and executing the protocol π) and the ideal world (with dummy players interacting with $\mathcal{S}$ and $\mathcal{F}$) with a significant advantage. Recall that the adversary is adaptive, which means that it can corrupt any player at any time during the execution of the protocol.

The main idea is that we use an extractable and equivocable commitment, which allows the simulator (while simulating the user) to be able to open it to any credential. This will be useful in case of adaptive corruptions. Indeed, in this case, if the credentials were correct, the simulator can adapt them and the randomness so that they seem to belong to the adequate language. From the server's side, the extractability of the commitment enables the simulator to know whether it has to send the correct message (obtained from the functionality in case the user is corrupted) or not.

This leads to the following simulator:

- when the simulator receives a Send-message from the ideal functionality, it knows than an honest sender has sent a pre-flow. It thus generates a key pair $(\mathsf{pk}, \mathsf{sk}) \overset{\$}{\leftarrow} \mathsf{KeyGen}(1^{\mathfrak{K}})$ and sends pk as a pre-flow.
- when the simulator receives a $\mathtt{Receive}$-message from the ideal functionality, it knows than an honest user has sent a pre-flow. It also has received a pre-flow pk (from an honest or a corrupted sender). It then generates an equivocable commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1) \overset{\$}{\leftarrow} \mathsf{Com}^{\ell}(\mathsf{crs}, \mathsf{Cred}_i, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ with $\ell = (\mathsf{sid}, \mathsf{ssid}, P_i, P_j)$ and a ciphertext $c \overset{\$}{\leftarrow} \mathsf{Encrypt}_{\mathsf{cpa}}(\mathsf{pk}, J)$ where S is a random value.
- when it simulates an honest server who receives these values $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1)$ and c from a corrupted user, it decrypts the ciphertext c as J, and computes $S = F(J)$. Next, it extracts the committed values (Cred_i), which it uses to send a $\mathtt{Receive}$-message to the ideal functionality.
- when it simulates an honest user receiving (hp_P, N_P) from a corrupted server, it computes $K \leftarrow \mathsf{ProjHash}(\mathsf{hp}_P, (\mathcal{L}_P, \ell, [\boldsymbol{C}]_1, [\boldsymbol{R}]_2), [\boldsymbol{\Pi}]_1)$ and gets $M \leftarrow S \oplus K \oplus N_P$. It uses this value in a Send query to the ideal functionality.

- when it simulates an honest server and receives a `Received`-message from the ideal functionality, giving it M sent to the corrupted user, it proceeds with this value M.
- when it simulates an honest server facing an honest user (the simulator itself), on the behalf of which it generated the commitment $([\boldsymbol{C}]_1, [\boldsymbol{R}]_2, [\boldsymbol{\Pi}]_1)$ and the ciphertext c, it uses $M = 0$ and chooses S at random (instead of computing it honestly with $F(J)$). This value S will be adapted during the simulation in case of corruption afterwards (which gives it the message M received by the user in case his credentials comply with the policy), so that $M = S \oplus K \oplus N_P$.