École normale supérieure
Département d'Informatique

Université Paris 7
Denis Diderot

Automorphic Signatures and Applications

PhD thesis

Georg Fuchsbauer

13 October 2010

Abstract

We advocate modular design of cryptographic primitives and give building blocks to achieve this efficiently. This thesis introduces two new primitives called *automorphic signatures* and *commuting signatures*, and illustrates their usefulness by giving numerous applications.

Automorphic signatures are digital signatures satisfying the following properties: the verification keys lie in the message space, messages and signatures consist of elements of a bilinear group, and verification is done by evaluating a set of pairing-product equations. These signatures make a perfect counterpart to the efficient proof system by Groth and Sahai (EUROCRYPT '08). We provide practical instantiations of automorphic signatures under appropriate assumptions and use them to construct the first efficient round-optimal blind signatures. By combining them with Groth-Sahai proofs, we moreover give practical instantiations of various other cryptographic primitives, such as fully-secure group signatures, non-interactive anonymous credentials and anonymous proxy signatures. To do so, we show how to transform signature schemes whose message space is a group to a scheme that signs arbitrarily many messages at once.

Verifiable encryption allows to encrypt a signature and prove that the plaintext is valid. Commuting signatures extend verifiable encryption in multiple ways: A signer can encrypt both signature and message and prove validity. More importantly, given a ciphertext, a signer can create a verifiably encrypted signature on the encrypted (unknown) message, which leads to the same result as signing the plaintext and verifiably encrypting the message and the signature; thus, signing and encrypting commute. We instantiate commuting signatures by combining our automorphic signatures with the encryption and proof system by Groth and Sahai, of which we prove a series of useful properties.

As an application, we give an instantiation of *delegatable anonymous credentials*, a powerful primitive introduced by Belenkiy et al. (CRYPTO '09). Our instantiation is arguably simpler than theirs and it is the first to provide *non-interactive* (and thus concurrently secure) issuing and delegation, which is standard for non-anonymous credentials. Moreover, the size of our credentials and the cost of verification are less than half of those of the only previous construction, and efficiency of issuing and delegation is increased even more significantly. All our constructions are proved secure in the standard model under non-interactive assumptions.

Contents

Chapter 1

Introduction

Contents

There are two antagonal approaches to the design of advanced cryptographic systems. The *modular* approach splits the task into smaller parts and treats them separately: after implementing them and proving that they satisfy the desired properties, the building blocks can be assembled to a more complex scheme on a more abstract level. An *ad hoc* approach is to solve the problem as a whole. The main advantage of the latter is that in practice it yields more efficient results. However, due to the loss of conceptual simplicity this usually results in more complex security proofs, as one cannot draw upon existing results.

Modular design, on the other hand, allows to build complex cryptographic primitives layer by layer, and to follow an intuitive approach, which facilitates both their design and their analysis, and enables making use of existing results. Ideally, one simply assembles existing instantiations and analyzes the security of the result by treating the building blocks as black boxes. This however requires that these blocks *fit* together. A drawback is that the philosophy of clean modular design might be to the disadvantage of efficiency—which is acceptable if the goal is to merely show feasibility of a new concept (below, we give the example of a strong security model for dynamic group signatures), but is undesirable to instantiate practical schemes.

We contribute to bridging the gap between modular design and efficiency by proposing the first signature scheme that is compatible with the (to date) only efficient standard-model zero-knowledge proof system in a comprehensive sense. Digital signatures and non-interactive zero-knowledge (NIZK) proof systems [BFM88] are the core of many cryptographic primitives providing means of identification or authentication, and at the same time anonymity. A classical example are group signatures [Cv91]: they allow members which were enrolled by a group manager to sign on behalf of a group without revealing their identity. To prevent misuse, anonymity can be revoked by an authority. Another primitive are anonymous credentials [Cha85], by which a user can prove that she holds a certain credential, while remaining anonymous. Blind signatures [Cha82] were introduced for electronic cash to prevent the linking of a coin to its spender, and are also used in electronic voting systems, where anonymity is indispensable. Other examples are verifiably encrypted signatures [BGLS03] and group encryption [KTY07].

Our new signature scheme provides means to instantiate such primitives in a modular way retaining conceptual simplicity, and at the same time yielding efficiency, sometimes even outperforming existing constructions as we show for *delegatable* anonymous credentials.

Group Signatures. As an example for modular protocol design let us consider the *BSZ model* for *dynamic group signatures* by Bellare et al. [BSZ05]. In their model there are 3 types of protagonists: the *issuer*, the *opener* and *group members*. The system is set up by computing a *group public key*, an *issuer key* and an *opening key*. To become a member of the group, a user runs the *group-joining protocol* with the issuer (holding the issuer key) at the end of which the user obtains a private signing key. Using this key, the member can make a signature on behalf of the group, which is verified using the group public key. In case of dispute, the opening authority can *open* group signatures: on input the opening key and a signature, an algorithm outputs the identity of the signer and a proof of correct opening.

The first security requirement is *anonymity*, intuitively meaning that signatures of different members are indistinguishable. Unforgeability is separated in two notions: *traceability* states that every valid signature can be opened to a registered user; and *non-frameability* requires that no coalition, possibly comprising issuer and opener, can produce a signature and a proof of correct opening that wrongfully accuse an honest user.

To show feasibility of their strong model, Bellare et al. give the following generic construction. Assume the existence of a signature scheme, an encryption scheme and non-interactive zero-knowledge proofs (see the next section), which follows from existence of trapdoor permutations. The setup produces a pair of verification and signing keys and a pair of encryption and decryption keys. The group public key consists of the verification and the encryption key and a common reference string for the NIZK. The issuer key is defined as the signing key and the opening key as the decryption key. When joining the group, a user produces a personal signature key pair and gets a certificate (i.e., a signature under the issuer key) on the verification key from the issuer. A member produces a group signature by first signing the message with her personal signing key, and then encrypting her certificate, her verification key, and the signature on the message. The group signature consists of these ciphertexts completed by a NIZK proof that the certificate and the signature in the plaintext are valid. A group signature is opened by decrypting the contained ciphertext: the revealed verification key identifies the user, and the signature acts as an unforgeable proof.

The fact that a signature is a ciphertext and a NIZK proof that leaks no information guarantees user anonymity. Traceability follows by unforgeability of the issuer's certificates and non-frameability by unforgeability of the user's signatures. Due to the compatibility with an existing encryption and proof scheme, our signature scheme allows to efficiently instantiate the building blocks; the security results of the generic construction then carry over directly to the resulting practical scheme.

For a long time the only efficient ways to instantiate privacy-preserving primitives was to either rely on the random-oracle heuristic [BR93] for NIZK—or to directly use *interactive* assumptions (like the LRSW assumption [LRSW00] and its variants, or "one-more" assumptions, as in [BNPS03]). Group signatures have been instantiated in the random-oracle model for example by Boneh et al. [BBS04] using Boneh-Boyen signatures [BB04] and by Camenisch and Lysyanskaya [CL04], who devise their own scheme relying on the LRSW assumption.

Due to a series of criticisms starting with [CGH98] more and more practical schemes are being proposed and proved secure in the *standard model* (i.e., without random oracles) and under *falsifiable* (and thus non-interactive) assumptions [Nao03]. All results of this work satisfy these two criteria.

1.1 Automorphic Signatures

1.1.1 Motivation

An *NP language* $\mathcal{L}$ is defined by a polynomial-time relation $R(\cdot,\cdot)$ as $\mathcal{L} := \{y \mid \exists w : R(y,w) = 1\}$, where y is called the *statement* and w is called the *witness*, since it allows to efficiently verify that $y \in \mathcal{L}$. A *non-interactive proof system* [GMR85] for such a language allows to prove that $y \in \mathcal{L}$. Such a system consists of a setup algorithm outputting a *common reference string* (CRS), a *prover* that has as input the CRS and (y,w) with $R(y,w) = 1$ and produces a proof π, and a *verifier* that on input the CRS, y and π outputs a decision bit. Completeness states that an honestly computed proof is accepted by the verifier and soundness requires that no prover can convince the verifier of a false statement.

We will use two security requirements for proof systems. *Witness indistinguishability* means that the proof does not reveal which witness the prover used to produce it. (Non-interactive) *zero knowledge* [BFM88] intuitively means that the proof does not leak anything more than validity of the statement ($y \in \mathcal{L}$). This is formalized by requiring that there exist a *simulation setup* which besides a CRS that is indistinguishable from a regular CRS outputs a *trapdoor*. Using this trapdoor, a *simulator* can produce a proof of a statement $y \in \mathcal{L}$ without using a witness. This proof must be indistinguishable from a proof output by the regular prover; formally: no adversary that, after receiving a CRS, outputs a pair (y,w) with $R(y,w) = 1$, and gets either a real proof or a simulated proof, can decide which type of proof it got. If the proofs are indistinguishable even if the adversary receives the trapdoor, the system is *composable* zero-knowledge [Gro06].

The languages we use in this thesis are typically of the following form: a ciphertext is in the language if it encrypts a valid signature; or the language is that of triples of encryptions of a signature verification key, a message and a valid signature on it. Note that such languages are NP since given as a witness the plaintexts and the randomness used to encrypt them, we can efficiently verify the validity of the statement.

The Groth-Sahai Proof System. Until recently, most of the practical instantiations relying on non-interactive zero-knowledge (and witness-indistinguishable) proofs use ad hoc constructions in the random-oracle model, since the generic standard-model instantiations are by far too inefficient. To instantiate encryption and signature schemes, groups with a bilinear map (pairing) turned out to be an attractive tool to achieve efficiency. In [GS08], Groth and Sahai propose *efficient* zero-knowledge proofs for a large class of statements over bilinear groups, which already found use in many implementations [CGS07, Gro07, GL07, BCKL08, CCS09, BCKL09, BCC$^+$09, 4]. They start by constructing witness-indistinguishable (WI) proofs of satisfiability of various types of equations: given a witness of satisfiability, one makes *commitments* to its values and then constructs proofs which assert that the committed values satisfy the equations. As already observed in [Gro06], the most interesting and widely used type is the following: pairing-product equations (PPE) whose variables are elements of the bilinear group (cf. Sect. 2.4). A PPE consists of products of pairings applied to the variables and constants from the group. Since the employed commitments to group elements are extractable, and are thus encryptions, the resulting proofs actually constitute *proofs of knowledge* [DP92].

To efficiently implement the generic construction for the BSZ model of group signatures from [BSZ05], Groth [Gro07] instantiates encryption and proofs of plaintext validity with the Groth-Sahai WI proof system. Extractability of the commitments serves two purposes: first, it lets the opener extract the user's verification key and thereby trace the signer (the commitments are thus used as encryptions that can be decrypted with the extraction key); second, it makes it possible

to reduce unforgeability of group signatures (i.e., traceability and non-frameability) directly to the unforgeability of the underlying signatures. For the Groth-Sahai methodology to be applicable, Groth gives certification and signing schemes such that certificates, signature verification keys and signatures (i.e., the components that need to be hidden) are group elements whose validity is verified by evaluating PPEs.[1] (cf. Sect. 6.1.3).

Signatures and the Groth-Sahai Proof System. The first practical schemes to use Groth-Sahai-like proofs were the group signatures by Boyen and Waters [BW06, BW07], who independently developed their proofs using techniques from [GOS06]. They require weakly secure[2] signatures whose components and messages can be encrypted (committed to) and proved to be valid. To define certificates lying in the bilinear group, they modify the weak Boneh-Boyen signatures [BB04], which consist of one group element and whose messages are scalars: instead of giving the scalar directly, they give it as an exponentiation of two different group generators. The security of their construction holds under a variant of the *strong Diffie-Hellman assumption* (SDH) [BB04] called *hidden SDH* (HSDH). (See Sect. 2.3 for the definitions of the assumptions.)

Belenkiy et al. [BCKL08] apply the Boneh-Boyen [BB04] transformation "from weak to strong security" to the Boyen-Waters scheme. They thereby obtain fully secure signatures, at the price of introducing a "very strong assumption" (according to [BCC$^+$09]) they call *triple Diffie-Hellman*. Their signatures consist of group elements, yet the messages are scalars. To construct anonymous credentials, they make commitments to a message and a signature on it and prove that their content is valid using Groth-Sahai proofs. Since from the employed commitments only group elements can be extracted efficiently, they are obliged to define *f-extractability*, meaning that only a function of the committed value can be extracted. This entails stronger security notions ("F-unforgeability") for the signature scheme in order to prove security of their construction.

In the abovementioned group signatures from [Gro07] this drawback is avoided by designing the key-certification scheme so that all committed values are group elements. The key certification is thus different from the signature scheme whose keys are certified. Moreover, the certificate-verification key is an element of the *target* group. As opposed to standard group signatures, in hierarchical group signatures [TW05] or anonymous proxy signatures [1], or more generally, to instantiate certification *chains*, verification keys are not only certified once, but must also serve to certify other keys. The message space must thus contain the verification keys. If we want to apply the Groth-Sahai methodology to "anonymize" such schemes and prove unforgeability by reducing it to the security of the underlying signatures, everything has to be in the bilinear group.

We identify the all-purpose building block to efficiently instantiate privacy-related primitives in combination with Groth-Sahai proofs as a digital signature scheme with the following properties:

- the scheme is existentially unforgeable against chosen-message attacks;
- the verification keys lie in the message space; and
- the messages and signatures are elements of a bilinear group and the signature-verification equations are PPEs;

Moreover, the scheme should be efficient (messages and signatures consist of a small number of group elements and verification requires a small number of pairing computations). We call such a

[1] The certified signatures defined by Ateniese et al. [ACHM05] satisfy these properties as well (and they can be completely randomized). The certificates are (a variant of) CL signatures [CL04] on the user's secret key; certification is thus an interactive protocol. Moreover, their construction strongly relies on *interactive* (thus non-falsifiable) assumptions, such as the strong LRSW [ACdM05] assumption.

[2] Throughout this thesis we call a signature scheme *weakly secure* if an adversary getting signatures on *random* messages cannot produce a signature on a new message.

scheme an *automorphic signature*, as it is able to sign its *own* keys and verification preserves the *structure* of keys and messages, which makes it perfectly suitable to be combined with Groth-Sahai proofs. We believe that working with group elements enables a modular approach of combining signatures with Groth-Sahai proofs, and automorphic signatures are the building block tailored to do so. As demonstrated in Chapter 6, they yield straightforward efficient implementations of generic constructions of a variety of primitives, by simply plugging in concrete schemes for generic ones.

We note that a scheme in [Gro06] based on the *decision linear assumption* [BBS04] can be considered automorphic, but should rather be regarded as a proof of concept due to its inefficiency (a signature consists of hundreds of thousands of group elements), whereas we give practical-level efficiency under reasonable assumptions. Independently of our work, Cathalo, Libert and Yung [CLY09] gave a practical signature scheme whose messages *and* signatures are group elements. However, as for the certification scheme from [Gro07], the verification keys contain an element from the target group. We now present two more primitives of which we will give the *first* efficient instantiation—using automorphic signatures.

Round-Optimal Blind Signatures. Blind signatures, introduced by Chaum [Cha82], allow a user to obtain a signature on a message such that the signer cannot relate the resulting message/signature pair to the execution of the signing protocol. They were formalized by [JLO97, PS00] and practical schemes without random oracles have been constructed in e.g. [CKW04, KZ06, Oka06, KZ08]. However, all these schemes require more than one round (i.e., two moves) of communication between the user and the signer to issue a blind signature. This is even the case for most instantiations in the random-oracle model, an exception being Chaum's scheme proved secure in [BNPS03] under an interactive assumption.

In [Fis06], Fischlin gives a generic construction of *round-optimal* blind signatures in the common-reference string (CRS) model: the signing protocol consists of one message from the user to the signer and one response by the signer. This immediately implies *concurrent* security, an explicit goal in other works such as [HKKL07]. Up to now, a practical instantiation of round-optimal blind signatures in the standard model has been an open problem.

Anonymous Proxy Signatures. Proxy signatures allow the delegation of signing rights; they were introduced by [MUO96] and later formalized in [BPW03, SMP08]. In [1] we introduced *anonymous* proxy signatures, which unify (multi-level) proxy signatures and group signatures by guaranteeing anonymity to the proxy signer and intermediate delegators.

They enable users ("*original delegators*") to delegate others to sign on their behalf; the latter, called *delegatees*, can furthermore re-delegate the received rights to other users. Anonymity ensures that proxy signatures do not reveal who signed and who re-delegated; however, they guarantee that there exists a delegation chain from the original delegator to the proxy signer. As for group signatures, an algorithm to revoke anonymity is provided to deter from misuse. Due to consecutiveness of delegation, this primitive also models hierarchical group signatures satisfying a security model generalizing the one of [BSZ05]. The only prior concrete instantiation of anonymous proxy signatures was given in [3], where we used Groth-Sahai-like proofs; it is however fairly impractical and relies on a new type of assumption.

1.1.2 Instantiations and Applications

In Chapter 4 we give two concrete instantiations of automorphic signatures and show them to be strongly unforgeable under chosen-message attack (Sect. 4.1). The first one relies on an assumption

which we introduced in [4]: the *double hidden SDH* assumption (DH-SDH) is a variant of SDH in the flavor of HSDH in symmetric bilinear groups ("Type-1" in the terminology of [GPS08]). As also pointed out by [GSW09a], the most efficient instantiation of Groth-Sahai (GS) proofs is the one in *asymmetric* bilinear groups ("Type-3") based on SXDH (cf. Sect. 2.2). In order to construct automorphic signatures over these groups, we define a variant of DH-SDH in asymmetric groups, called ***A***DH-SDH and prove it secure in the generic group model [Sho97]. Lastly, we give a new type of flexible CDH assumption, which is weaker than all previous versions such as [LV08]. Together with ADH-SDH it implies strong unforgeability of our second automorphic-signature instantiation in asymmetric bilinear groups. The scheme can be combined with the SXDH-instantiation of GS proofs and its signatures consist of only 5 group elements. We insist that all our assumptions are non-interactive and falsifiable [Nao03], and hold in the generic group model.

In Sect. 4.2, we use our schemes to give the first efficient instantiation of round-optimal blind signatures. The blind signature and the user message are of order 30 group elements (depending on the instantiation of the employed GS proofs) and the signer message consists of 5 elements. They can be based on either DH-SDH or ADH-SDH, the latter leading to a scheme that is automorphic itself, which makes it especially suitable for our applications. In Sect. 4.3, we give a generic transformation of a signature scheme whose message space is an algebraic group and contains the verification keys to one that signs vectors of arbitrary length. Our transformation preserves the structure of verification; thus applied to an automorphic scheme the resulting scheme is automorphic.

Applications of Automorphic Signatures. In Chapter 6 we give efficient black-box constructions of cryptographic primitives by combining GS proofs and automorphic signatures. We first discuss the generic construction of round-optimal blind signatures in the common-reference-string model by [Fis06]—although the direct (non-black-box) construction in Sect. 4.2 yields a more efficient result.[3] We note that in [11] we expanded on these ideas to give the first instantiation of *fair* blind signatures *without random oracles*; these are blind signatures where an authority can revoke anonymity: the authority can trace a signature to the user it was issued to and link an issuing to the issued signature.

In Sect. 6.1.2 and 6.1.3, we use automorphic signatures to build group signatures satisfying the BSZ model, and revisit the construction of non-interactive anonymous credentials of [BCKL08]; in particular, we achieve actual message extractability and give an efficient credential-issuing protocol. We then present the first efficient instantiation of anonymous proxy signatures (APS) in the standard model. We use automorphic signatures to certify public keys, so delegation is done by simply signing the delegatee's public key. An anonymous proxy signature is a Groth-Sahai (GS) proof of knowledge of a certification chain that starts at the original delegator and ends at the message.

We then strengthen the model for APS by enhancing the anonymity guarantees in Sect. 6.2.5. We first revise delegation so that intermediate delegators remain anonymous to the delegatee whereas the generic construction in [3] only provides anonymity w.r.t. the verifier. Moreover, we give a protocol for *blind delegation*: a user can be delegated to without revealing her identity. These enhancements do not affect the signature size, which grows linearly in the number of delegations (which is optimal, since the signature must contain opening information).

Recently, Belenkiy et al. [BCC+09] (BCCKLS) introduced *delegatable anonymous credentials*. They also provide mechanisms enabling users to prove possession of certain rights while remaining anonymous; and they consider re-delegation of received rights. (We discuss this primitive in more detail in the next section.) Similarly to the construction of APS, a delegatable credential consists

[3] The *ad hoc* construction yields thus a better result than the *modular* construction. In Chapter 5 the blind signature is however superseded by a more powerful primitive, which can then be used as a black-box building block for more complex protocols.

of a chain of certificates that is encrypted and proved valid. The core protocol of the BCCKLS instantiation lets a user obtain a proof of knowledge of an authenticator on her *secret* key, without revealing the identity of neither the signer nor the user. This imposes interactivity of the delegation process, while (non-blind) delegations for APS are non-interactive, even when delegators remain anonymous. (We show how to achieve delegatee anonymity at the expense of non-interactivity). Besides, the BCCKLS model only deals with authentication rather than signing, and does not provide tracing mechanisms.

We will show that using our second new primitive called *commuting signatures*, whose instantiation uses automorphic signatures and which we introduce in the next section, we can combine the best of both instantiations: non-interactive delegation and anonymity of *both* delegator and delegatee w.r.t. each other.

1.2 Commuting Signatures and Verifiable Encryption

1.2.1 Motivation and Definition

Verifiable Encryption. A verifiably encrypted signature scheme [BGLS03, RS09] enables a signer to make a digital signature on a message, encrypt the signature under a third party's encryption key, and produce a proof asserting that the ciphertext contains a valid signature. Suppose the message is only available as an encryption. The signer cannot make a signature on the plaintext, as this would contradict the security of the encryption scheme (given two messages and the encryption of one of them, a signature on the plaintext could be used to decide which message was encrypted). However, the following does not seem a priori impossible: given a ciphertext, instead of directly making a signature on the plaintext, the signer could produce a verifiable encryption of a signature on the plaintext.

We show that—surprisingly—such a functionality is feasible and moreover give a practical instantiation of it. We then use this new primitive to build the first non-interactively delegatable anonymous credential scheme: given an encrypted public key, a delegator can make a verifiably encrypted certificate on the key, which acts as credential.

Delegatable Anonymous Credentials. Access control that respects users' privacy concerns is a challenging problem in security. To gain access to resources, a participant must prove to possess the required credential issued by an authority. To increase manageability of the system, the authority usually does not issue credentials directly to each user, but relies on intermediate layers in the hierarchy. Belenkiy et al. [BCC$^+$09] give the following example: a system administrator issues credentials for webmasters to use his server. The latter are entitled to create forums and delegate rights to moderators, who in turn can give posting privileges to users.

Web-based social network services are enjoying a huge popularity and represent another area of application for credentials. Registered users can be given credentials to access services, which they delegate to introduce and recommend friends and friend of friends. The recent rise of concern about protection of privacy in such networks motivates *anonymous* credentials: a user can obtain a credential and prove possession of it without revealing neither her identity nor that of the user that delegated it to her.

In the real world (non-anonymous) delegation of rights is usually realized by certifying the public key of the delegated user. Consecutive delegation leads to a credential chain, consisting of public keys and certificates linking them, starting with the *original issuer* of the credential and

ending with a user, say Alice. To delegate her credential to Bob, Alice simply extends the length of the chain by one by appending a certificate on Bob's public key, produced with her secret key.

Anonymous credentials [Cha85, Dam90, LRSW00, Bra99, CL01, CL02, CL04, BCKL08] aim to provide a functionality similar to certificates while at the same time not revealing information about the user's identity when obtaining or showing a credential. However, the goal of reconciling delegatability and anonymity remained elusive—until recently. Chase and Lysyanskaya [CL06] construct a first delegatable anonymous credential scheme; yet, the size of a credential is *exponential* in its length (i.e., the number of delegations), which makes them impractical. A breakthrough was made in [BCC$^+$09], where Belenkiy et al. (BCCKLS) introduce a new approach using a non-interactive zero-knowledge (NIZK) proof system with *randomizable* proofs: given such a proof, anyone can transform it to a new proof that cannot be linked to the original one. A credential is a non-interactive *proof of knowledge* of a certification chain that can be randomized before being delegated or shown; this guarantees anonymity and unlinkability.

The functionality of the system can be sketched as follows: each user holds a secret key which she uses to produce multiple pseudonyms Nym, which cannot be linked to each other. A user A can be known to user O as $Nym_A^{(O)}$ and to B as $Nym_A^{(B)}$. Given a credential issued by O for $Nym_A^{(O)}$, A can transform it into a credential for $Nym_A^{(B)}$ and show it to B. Moreover A can delegate the credential to user C, known to A as $Nym_C^{(A)}$. C can then show a credential from O for $Nym_C^{(D)}$ to user D (without revealing neither $Nym_A^{(C)}$ nor $Nym_C^{(A)}$), or redelegate it, and so on.

Delegation preserves anonymity, i.e., delegator and delegatee learn nothing more about each other than their respective pseudonyms. In the instantiation of [BCC$^+$09], the delegation protocol is fairly complex and highly interactive—in contrast to (non-anonymous) credentials, where it suffices to know a user's public key in order to issue or delegate a credential to her. We overcome this shortcoming by giving an instantiation of the BCCKLS model that enables *non-interactive* delegation: pseudonyms are encryptions of a user public key, and given a pseudonym Nym, the delegator can produce a credential for the holder of Nym without any interaction, by making a proof of knowledge of a signature on the public key given to her as an encryption Nym. This can be achieved using commuting signatures and verifiable encryption, the latter representing a proof of knowledge of a signature.

Note that a non-interactive delegation protocol immediately yields security against concurrent attacks where an adversary might simultaneously run protocols for delegating or being delegated credentials with honest users. This was not considered in the BCCKLS model. An interesting property of our instantiation is that the delegation is not only a non-interactive protocol, but the delegator produces a *ready* credential for the user's pseudonym without *any* interaction of the user.

Finally, we note that, as for the BCCKLS instantiation, abuse prevention mechanisms such as anonymity revocation [CL01] or limited show [CHK$^+$06] can be added to our construction.

Commuting Signatures and Verifiable Encryption. Our main building block to instantiate non-interactively delegatable anonymous credentials is a new primitive we call *commuting signature* which we sketch in the following and formally define in Sect. 5.2. It combines a digital signature scheme, an encryption scheme and a proof system with the following properties: given a verification key, a message and a signature on it valid under the key, we can encrypt any subset of {key, message, signature}, and make a proof that the plaintexts constitute a triple of a key, a message and a valid signature. We also require that the proof does not leak any more information about the encrypted values besides validity.

For consistency with our instantiation using the Groth-Sahai methodology [GS08], we will say *commitment* instead of *encryption*. Note that the commitments we use are *extractable*, and therefore constitute an encryption scheme (see Sect. 2.2.1). We denote committing to signatures by Com and committing to messages by $\mathsf{Com}_{\mathcal{M}}$. A proof for a committed signature is denoted $\widetilde{\pi}$, and a proof

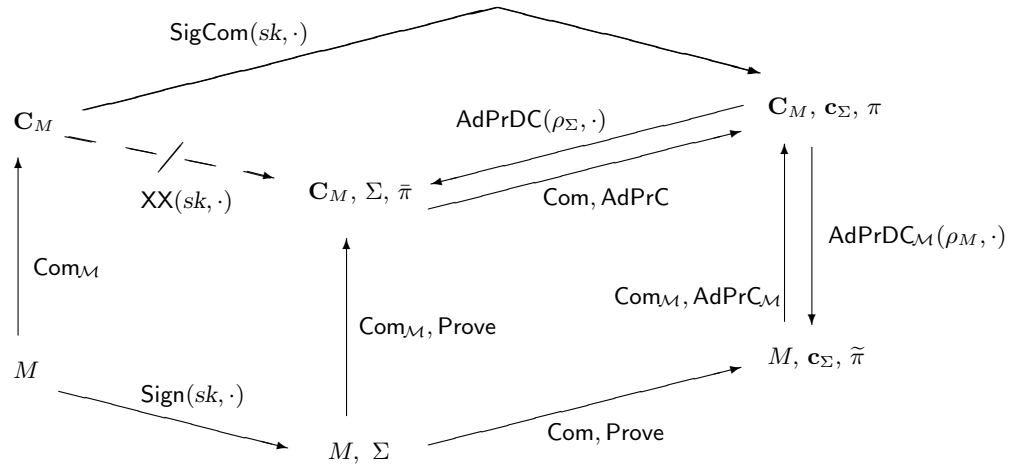

Figure 1.1: Commuting signatures

for a committed message by $\bar{\pi}$. If both are committed, we simply write π, and if the **v**erification key is committed too, we write $\widehat{\pi}$.

Besides allowing to prove validity of committed values, a commuting signature scheme provides the following additional functionalities (sketched in Figure 1.1). Note that none of them requires knowledge of the extraction (decryption) key.

SigCom. Given a commitment $\mathbf{C}_M$ to a message M and a signing key sk, SigCom produces a commitment $\mathbf{c}_\Sigma$ to a signature Σ on M under sk, and a proof π that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$.

AdPrC ("adapt proof when committing"). Given a commitment $\mathbf{C}_M$ to M, a signature Σ on M and a proof $\bar{\pi}$ of validity of Σ on the content of $\mathbf{C}_M$, we can make a commitment $\mathbf{c}_\Sigma$ to Σ using randomness ρ_Σ and run algorithm AdPrC on $\mathbf{C}_M, \Sigma, \rho_\Sigma$ and $\bar{\pi}$. Its output is a proof π that the content of $\mathbf{c}_\Sigma$ is a valid signature on the content of $\mathbf{C}_M$.
AdPrDC ("adapt proof when decommitting") does the converse: given a committed message $\mathbf{C}_M$, a committed signature $\mathbf{c}_\Sigma$ together with the used randomness ρ_Σ, and a proof π for $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$, AdPrDC outputs a proof $\bar{\pi}$ of validity of the signature Σ on the committed message.

AdPrC$_\mathcal{M}$. Analogously we define algorithms for proof adaptation when committing and decommitting to the *message*. Given a message M, a commitment $\mathbf{c}_\Sigma$ to a signature on M and a proof of validity $\widetilde{\pi}$, AdPrC$_\mathcal{M}$ transforms the proof to the case when the message is committed as well. AdPrDC$_\mathcal{M}$ is given commitments $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$ to a signature and a message M, the randomness ρ_M for $\mathbf{C}_M$ and a proof π. It adapts π to a proof $\widetilde{\pi}$ that the content of $\mathbf{c}_\Sigma$ is a valid signature on M.

AdPrC$_\mathcal{K}$. We can also adapt proofs when committing or decommitting to the *verification key*. Given commitments $\mathbf{C}_M$ and $\mathbf{c}_\Sigma$ to a message and a signature, a proof of validity π, the verification key vk and randomness ρ_{vk}, AdPrC$_\mathcal{K}$ outputs a proof $\widehat{\pi}$ that the content of $\mathbf{c}_\Sigma$ is a signature on the content of $\mathbf{C}_M$ valid under the key vk given as a commitment $\mathbf{c}_{vk}$ with randomness ρ_{vk}. AdPrDC$_\mathcal{K}$ is given $(vk, \rho_{vk}, \mathbf{C}_M, \mathbf{c}_\Sigma, \widehat{\pi})$ and adapts the proof $\widehat{\pi}$ for $(\mathbf{c}_{vk}, \mathbf{C}_M, \mathbf{c}_\Sigma)$, where $\mathbf{c}_{vk}$ is a commitment to vk with randomness ρ_{vk}, to a proof for $(vk, \mathbf{C}_M, \mathbf{c}_\Sigma)$.

We require that committing, signing and the functionalities above *commute* with each other, that

is, it does not matter in which order we execute them; e.g., signing a message, committing to the message and the signature and proving validity yields the same as committing to the message and then running SigCom. Thus, the diagram in Figure 1.1 commutes. Note that due to the argument given in the beginning of Sect. 1.2.1 there cannot exist a functionality XX that is given a commitment $\mathbf{C}_M$ to a message M and a secret key sk, and outputs a signature Σ on M.

Besides verifiably encrypted signatures (Sect. 5.1) and blind signatures (Sect. 5.2.1), commuting signatures also immediately yield *CL signatures* [CL01, CL02, CL04], an important building block for protocols providing privacy. They allow a user to obtain a signature on a committed value from a signer by running a protocol *Issue*. The user can then make a proof of knowledge of that signature which is verifiable given the commitment. SigCom provides a non-interactive issuing, which directly gives the user a proof of knowledge of such a signature.

1.2.2 Instantiations

Instantiating Commuting Signatures. Our instantiation will be based on Groth-Sahai (GS) proofs and automorphic signatures, in particular the blind signature scheme from Sect. 4.2, which can be sketched as follows: the user *randomizes* the message by multiplying it with a random term, makes an (extractable) commitment to the message and the randomness and adds a witness-indistinguishable (WI) proof that the commitments contain the correct values. The signer receives the randomization, commitments and a proof and therefore learns nothing about the message. Using the randomized message, the signer fabricates a "pre-signature", from which, knowing the randomness, the user can retrieve an actual signature. To prevent the signer from linking the resulting signature to the signing session, the actual blind signature is a WI proof of knowledge (PoK) of the signature. The PoK consists of extractable commitments to the signature components and a WI proof that the committed values satisfy the signature verification equation on the message; in other words, the PoK is a verifiably encrypted signature. The commitments and WI proofs are instantiated with the GS methodology for committing to elements from a bilinear group and constructing proofs that they satisfy *pairing-product equations*.

We observe that the values the user sends to the signer can be seen as a commitment to (or an encryption of) the message. We show that this commitment can be used by the signer to *directly*—without the help of the user—construct a proof of knowledge of a signature on the committed message (that is, extractable commitments to the signature components and a proof that the committed values constitute a valid signature on the committed message). This is possible due to a series of properties of GS proofs we identify: the proofs are homomorphic w.r.t. the statement they prove; moreover, they are independent of parts of the statement and there are ways to "blindly" transform a proof for one statement to a proof for another statement (Sect. 5.3).

GS commitments are extractable, thus the extraction key acts as the decryption key and witness indistinguishability implies semantic security (cf. Sect. 2.2.1). We will use the notions *encryption* and *extractable commitment* interchangeably. An extractable commitment to a signature together with a proof of validity is a *verifiably encrypted signature* (VES) and can also be interpreted as a proof of knowledge of a signature (since by decryption, the signature can be extracted). Our instantiation of commuting signatures is given in Sect. 5.4.

Instantiating Delegatable Anonymous Credentials. Belenkiy et al. [BCC+09] show that Groth-Sahai proofs can be randomized and combine them with an authentication scheme for secret keys to construct delegatable credentials. A pseudonym *Nym* is a commitment to the user's secret key and a credential is a proof of knowledge of an authentication chain. Such a proof consists of commitments to secret keys, commitments to authenticators between the keys, and proofs of

validity. To issue or delegate, the issuer and the user jointly compute a proof of knowledge of an authenticator on the content of the user's pseudonym. In the case of delegation, the issuer prepends her own credential, which she randomizes before. The authors note that secret keys cannot be extracted from the commitments, and that an adversary against the authentication scheme must be allowed to ask for authenticators *on* as well as *under* the attacked key. They therefore give an *F-unforgeable certification-secure* authentication scheme.

We avoid this notion and interactivity of delegation by choosing a more modular approach. We replace the authenticators on secret keys by commuting signatures on public keys; in particular, the underlying signatures are automorphic, i.e., Groth-Sahai compatible with verification keys that lie in the message space—which is a requirement for delegation. A credential is then a chain of *public keys* and *certificates* (as in the non-anonymous case), which are all given as commitments completed with proofs of validity.

Commuting signatures enable non-interactive delegation (and issuing, which is a special case of delegation): given a pseudonym Nym_U (i.e., a commitment to the public key) of a user, the issuer can produce a commitment $\mathbf{c}_\Sigma$ to a signature on the committed user key and a proof π of validity using SigCom. In the case of issuing, the credential is $(\mathbf{c}_\Sigma, \pi)$ and is verified by checking π on the issuer's public key, Nym_U and $\mathbf{c}_\Sigma$. In the case of delegation the issuer randomizes her own credential $cred_I$, yielding a credential $cred_I{}'$ on her pseudonym Nym_I that is unlinkable to $cred_I$. Finally, running $\mathsf{AdPrC}_\mathcal{K}$, the issuer adapts the proof π (which is valid under her verification key) to a proof $\widehat{\pi}$ of validity of the signature contained in $\mathbf{c}_\Sigma$ on the content of the user pseudonym Nym_U under the content of the issuer's pseudonym Nym_I. The credential for the user is then $cred_I{}' \parallel Nym_I \parallel (\mathbf{c}_\Sigma, \widehat{\pi})$.

Comparing our Results to Previous Ones. Replacing the authenticators from [BCC+09], which consist of 11 group elements and are verified by 8 pairing-product equations (PPE), with the automorphic signatures from Sect. 7.3.1 (consisting of 5 group elements and satisfying 3 PPEs) more than doubles efficiency of the scheme. Moreover, in Sect. 7.4 we revise the approach to achieving simulatability of credentials. Groth and Sahai [GS08] show how to simulate *proofs of satisfiability* consisting of commitments and proofs for the committed values, which are both produced by the simulator. However, in order to simulate credentials, the simulator has to construct proofs for *given* commitments. Belenkiy et al. therefore double some of the commitments and provide proofs that the committed values are equivalent. We show that our equations can be *directly* simulated even if some of the commitments are fixed before simulation.

Most importantly, our delegation (and issuing) protocol outperforms theirs significantly (see Sect. 7.2.1 for a more detailed comparison): In the BCCKLS scheme, the issuer first sends a GS proof of knowledge of the first 6 signature components. The issuer and the user then run a two-party protocol to jointly compute the last component, using a homomorphic encryption scheme and interactive ZK proofs that blinding values are in the correct ranges. In our instantiation the issuer simply sends a GS proof of knowledge of our 5 signature components.

The assumptions under which the respective schemes are proven secure are the BB-CDH, the BB-HSDH, and the SXDH assumption for the instantiation of [BCC+09], whereas our instantiation is secure under ADH-SDH and SXDH (see Sect. 2.3 and 3.3). BB-HSDH, introduced in [BCC+09], and ADH-SDH are incomparable and both "hidden" variants of the *strong Diffie-Hellman* assumption [BB04]. They both fall in Boyen's generalized "Uber-Assumption" family [Boy08] and have the same generic security bound. We discuss similarities in Sect. 3.3.1.

We combined automorphic signatures with Groth-Sahai proofs in Sect. 6.2 to construct *anonymous proxy signatures*. This primitive is related to anonymous credentials in that it considers proving rights in an anonymous way; but it does not achieve mutual anonymity between the delegator and the delegated user. Note that if in our credential scheme we give the extraction key for

the commitments to a tracing authority, and if we define a *proxy signing algorithm* which works like delegation but produces a committed signature on a *clear* message rather than a committed user key, we get an instantiation of anonymous proxy signatures with *mutually anonymous* delegation.

We note that recently [AHO10] also introduced a signature scheme with messages and signatures from a bilinear group which are verified by pairing-product equations. While their initial scheme is not automorphic, they give an automorphic variant whose keys are however 33 times longer than ours, while signatures are 8 times longer.[4] They use their scheme to construct a blind signature, which follows Fischlin's [Fis06] approach, in which the user only gets a signature on a *commitment* of the message. It is thus not clear whether their scheme can be used to instantiate commuting signatures.

Overview. The part on commuting signatures and their application to delegatable credentials is structured as follows. Using the definitions of extractable commitments, randomizable witness-indistinguishable proofs, and compatible signatures from Sect. 2.2, we discuss how they can be combined to verifiably encrypted signatures in Sect. 5.1. In Sect. 5.2 we formally define commuting signatures and give some immediate black-box results, such as blind signatures. In Sect. 5.3 we state and prove 5 lemmas about properties of Groth-Sahai proofs which are used in Sect. 5.4, where we instantiate our commuting-signature scheme using Groth-Sahai proofs (Sect. 2.4) and automorphic signatures (Sect. 4.1). In Sect. 5.5, we give some complementary results; in particular, we describe how to extend commuting signatures when several messages are to be signed at once.

Chapter 7 is dedicated to delegatable credentials. In Sect. 7.1 we recall the model for delegatable anonymous credentials from [BCC$^+$09] and subsequently describe our instantiation providing non-interactive delegation in Sect. 7.2. We prove security and conclude with a comparison to the BCCKLS instantiation in Sect. 7.2.1. In Sect. 7.3 we give a variant of the automorphic signatures from Sect. 4.1 which enables a more efficient instantiation of delegatable credentials. In Sect. 7.4 we discuss some issues of simulatability of Groth-Sahai proofs that arise when they are used to instantiate delegatable credentials satisfying a simulation-based anonymity definition.

This Thesis. The results of this thesis appear in [6] and [8], with two exceptions: Sect. 2.4.5 contains results from [7] and Sect. 3.2 contains results from the full version of [4].

[4] Our keys and signatures are in $\mathbb{G}_1 \times \mathbb{G}_2$ and $\mathbb{G}_1^3 \times \mathbb{G}_2^2$ for asymmetric groups, whereas those of [AHO10] are in $\mathbb{G}^{50}$ and $\mathbb{G}^{28}$, respectively, where $\mathbb{G}$ is a (less efficient) symmetric bilinear group. We assumed $\mathbb{G}$ is of the same size as $\mathbb{G}_2$ while $\mathbb{G}_1$ elements have a representation half as long as $\mathbb{G}_2$ elements.

Chapter 2

Preliminaries

Contents

2.1 Notation

Since we are going to combine quite a few concepts we give a guideline on notation. We tried to stick to our framework, but deviated sometimes for the sake of consistency with other work such as Groth and Sahai's [GS08].

- Capital Roman letters denote elements of a bilinear group.[1] *Diffie-Hellman pairs* of group elements (see Sect. 2.3) are mostly two consecutive letters of the alphabet.
- Lower-case Roman letters denote integers. Mostly they correspond to the logarithm of the corresponding capital letter in a common basis, e.g. $M = G^m$.
- Greek letters denote the randomness used in the commitments to a group element that is denoted by the corresponding Roman letter, e.g. $\mathbf{c}_M = \mathsf{Com}(ck, M, \mu)$.
- $\mathcal{M}$, $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{C}$ denote spaces for messages, values, randomness and commitments, respectively; $\mathcal{E}$ denotes a class of equations, $\mathcal{H}$ a hash function, and $\mathcal{O}$ denotes an oracle. $\mathbb{G}$ denotes an algebraic group, $\mathbb{Z}$ denotes integers, $\mathbb{N}$ non-negative integers, and $\mathbb{R}$ denotes the reals. By $\mathbb{G}^*$ we denote $\mathbb{G} \setminus \{1\}$, i.e., the group without the neutral element; and analogously we define $\mathbb{Z}_p^* := \mathbb{Z}_p \setminus \{1\}$.

[1] Although generators of bilinear group are typically denoted by g, definitions like $X := g^x$ are very common in the literature. We think it is sensible to distinguish different types of variables by different types of letters, and accept that G being a generator rather than a group may look unusual at first.

- Special cases:
  - $\lambda \in \mathbb{N}$ denotes the security parameter
  - p denotes a prime (typically the group order), e denotes a bilinear map (pairing) and E denotes a generic equation;
  - ϕ and θ denote the components of Groth-Sahai proofs, whereas π denotes a generic proof;
  - $\vec{\mathbf{u}}$, $\vec{\mathbf{v}}$ with components u_{ij} and v_{ij} are the keys for Groth-Sahai commitments;
  - $\mathbf{c}, \mathbf{d}$ and $\mathbf{C}$ denote commitments;
  - $\mathfrak{t}_T$ denotes an element from the target group $\mathbb{G}_T$;
  - Γ and Z denote matrices in with entries γ_{ij} and z_{ij} in $\mathbb{Z}_p$, respectively.
- In general, we denote row vectors by bold letters and column vectors with arrows; e.g., $u_{11} \in \mathbb{G}$, $\mathbf{u}_1 \in \mathbb{G}^3$, and $\vec{\mathbf{u}} \in \mathbb{G}^{3\times 3}$.
- In a bilinear group, we denote the group operation by "$\cdot$". For vectors and matrices of group elements, "$\circ$" denotes applying the group operation componentwise.
- By ":=", we denote either a definition—with the definiens on the right-hand side (RHS) and the definiendum on the left-hand side (LHS)—or an assignment of the value on the RHS to the variable on the LHS.
- "$\leftarrow$" denotes a random assignment. If the RHS is a finite set it denotes choosing a value from it uniformly and assigning it to the LHS. If the RHS is a probabilistic algorithm it denotes choosing its random tape uniformly and assigning the outcome to the LHS.
- The concatenation of two bit strings s and t is denoted by $s\|t$. The empty string is denoted by ε. If $\lambda \in \mathbb{N}$ then 1^λ denotes the string of λ ones.
- Variables of cryptographic protocols are denoted in slanted font; e.g., *sk* is a signing key.
- Algorithms are denoted sans serif like Sign and are assumed to be probabilistic polynomial time (p.p.t.) or polynomial time (p.t.). We write e.g. $\Sigma \leftarrow \mathsf{Sign}(sk, M)$ if the algorithm is probabilistic. To make the internal randomness r explicit we write $\Sigma := \mathsf{Sign}(sk, M; r)$.
- A function $f\colon \mathbb{N} \to \mathbb{R}$ is called *negligible* if for all $c \in \mathbb{N}$ there exists $k_0 \in \mathbb{N}$ s.t. for all $k > k_0$: $|f(k)| < \frac{1}{k^c}$. Problems are called *hard* if no p.p.t. algorithm can solve them with non-negligible probability.

2.2 Formal Definition of Primitives

We formally define some cryptographic primitives from the literature.

2.2.1 Commitments

A (non-interactive) *randomizable extractable commitment scheme* **Com** is composed of the algorithms Setup, Com, RdCom, $\mathsf{ExSetup}$, Extr, and $\mathsf{WISetup}$, which define $\mathcal{V}$, the space of "committable" values, $\mathcal{R}$, the randomness space and $\mathcal{C}$ the space of commitments. Setup and $\mathsf{WISetup}$ output *commitment keys* *ck*, and Com, on inputs *ck*, a message $M \in \mathcal{V}$ and randomness $\rho \in \mathcal{R}$ outputs a commitment $\mathbf{c} \in \mathcal{C}$. $\mathsf{ExSetup}$ outputs (ck, ek), where *ck* is distributed as the output of Setup, and *ek* is the *extraction key*. We require the following:

- The scheme is *perfectly binding*, i.e., for any ck output by Setup and any commitment $\mathbf{c} \in \mathcal{C}$ there exists exactly one $M \in \mathcal{V}$ s.t. $\mathbf{c} = \mathsf{Com}(ck, M, \rho)$ for some ρ. Moreover, if $(ck, ek) \leftarrow \mathsf{ExSetup}$ then $\mathsf{Extr}(ek, \mathbf{c})$ extracts that value M from $\mathbf{c}$.
- The scheme is *computationally hiding*, in particular, the keys output by $\mathsf{WISetup}$ are computationally indistinguishable from those output by Setup; and they generate perfectly hiding commitments, i.e., for ck^* output by $\mathsf{WISetup}$ we have that for every $\mathbf{c} \in \mathcal{C}$ and $M \in \mathcal{V}$ there exists a $\rho \in \mathcal{R}$ s.t. $\mathbf{c} = \mathsf{Com}(ck^*, M, \rho)$.
- The scheme is *randomizable*, i.e., RdCom takes as input a key ck, a commitment $\mathbf{c}$ and fresh randomness $\rho' \leftarrow \mathcal{R}$ and outputs a *randomized* commitment $\mathbf{c}'$. If ρ' is chosen uniformly from $\mathcal{R}$ then $\mathbf{c}'$ is distributed as $\mathsf{Com}(ck, M, \rho)$ where ρ is picked uniformly from $\mathcal{R}$.

  We require that $(\mathcal{R}, +)$ is a group and that we have

$$\mathsf{RdCom}(ck, \mathsf{Com}(ck, M, \rho), \rho') = \mathsf{Com}(ck, M, \rho + \rho') \ .$$

A perfectly binding computationally hiding commitment scheme as defined above is actually a *lossy encryption scheme* [BHY09]. This is an encryption scheme with two key generation algorithms producing indistinguishable encryption keys. One outputs regular key pairs that lead to decryptable ciphertexts, whereas the other one returns "lossy" encryption keys, which lead to ciphertexts that are independent of the message. Lossy encryption schemes satisfy the standard IND-CPA definition of semantic security.[2]

We write Com also when we commit to a *vector* in $\mathcal{V}^n$: if $M = (M_1, \ldots, M_n)$ and $\rho = (\rho_1, \ldots, \rho_n)$ then $\mathsf{Com}(ck, M, \rho) := \big(\mathsf{Com}(ck, M_1, \rho_1), \ldots, \mathsf{Com}(ck, M_n, \rho_n)\big)$. Likewise, we define $\mathsf{Extr}(ck, (\mathbf{c}_1, \ldots, \mathbf{c}_n)) := \big(\mathsf{Extr}(ck, \mathbf{c}_1), \ldots, \mathsf{Extr}(ck, \mathbf{c}_n)\big)$.

2.2.2 Proofs for Committed Values

We define a proof system which allows to prove that committed values satisfy an equation. The proofs are constructed using the committed values and the used randomness. They are witness-indistinguishable, that is, they reveal nothing more than the fact that the committed values satisfy the equation. Moreover, given a proof for a set of commitments, the proof can be adapted to a randomization of the commitments (without knowledge of the committed values).

A *randomizable witness-indistinguishable proof system* **Proof** for a commitment scheme **Com** for a class $\mathcal{E}$ of equations consists of the algorithms Prove, Verify and $\mathsf{RdProof}$. Given values $M_1, \ldots, M_n \in \mathcal{V}$ satisfying an equation $\mathrm{E} \in \mathcal{E}$, the algorithm Prove, on input ck, a description of E, $(M_1, \ldots, M_n)$ and $\rho_1, \ldots, \rho_n \in \mathcal{R}$, outputs a proof π. On inputs ck, E, $\mathbf{c}_1, \ldots, \mathbf{c}_n$ and π, Verify outputs 0 or 1, indicating acceptance or rejection of a proof. We require that the system satisfies the following:

- *Completeness.* For all $(M_1, \ldots, M_n)$ satisfying $\mathrm{E} \in \mathcal{E}$, and all $\rho_1, \ldots, \rho_n \in \mathcal{R}$ we have

$$\mathsf{Verify}\big(ck, \mathrm{E}, \mathsf{Com}(ck, M_1, \rho_1), \ldots, \mathsf{Com}(ck, M_n, \rho_n), \mathsf{Prove}(ck, \mathrm{E}, (M_1, \rho_1), \ldots, (M_n, \rho_n))\big) = 1 \ .$$

[2] Consider the security game for *indistinguishability under chosen-plaintext attack* (IND-CPA) for encryption schemes. The challenger creates a pair of encryption and decryption key, and gives the encryption key to the adversary (who can thus make ciphertexts for plaintexts of his choice). The adversary outputs two messages, gets an encryption of one of them and has to guess which one. Replacing the key by a "lossy" encryption key output by $\mathsf{WISetup}$ is indistinguishable; and encryptions under a lossy key are independent of the message. Encryptions of different messages are thus indistinguishable.

- *Soundness.* Let $(ck, ek) \leftarrow \mathsf{ExSetup}$, $\mathrm{E} \in \mathcal{E}$, and $\mathbf{c}_1, \ldots, \mathbf{c}_n \in \mathcal{C}$. If there exists π such that $\mathsf{Verify}(ck, \mathrm{E}, \mathbf{c}_1, \ldots, \mathbf{c}_n, \pi) = 1$, then letting $M_i := \mathsf{Extr}(ek, \mathbf{c}_i)$, we have that $(M_1, \ldots, M_n)$ satisfy E.
- *Witness indistinguishability.* Let $ck^* \leftarrow \mathsf{WISetup}$, and $M_1, \ldots, M_n, M'_1, \ldots, M'_n \in \mathcal{V}$ be such that both $(M_1, \ldots, M_n)$ and $(M'_1, \ldots, M'_n)$ satisfy an equation $\mathrm{E} \in \mathcal{E}$. Let moreover $\rho_1, \ldots, \rho_n$, $\rho'_1, \ldots, \rho'_n \in \mathcal{R}$ be such that for all i, $\mathsf{Com}(ck^*, M_i, \rho_i) = \mathsf{Com}(ck^*, M'_i, \rho'_i)$. Then the outputs of $\mathsf{Prove}(ck^*, \mathrm{E}, (M_1, \rho_1), \ldots, (M_n, \rho_n))$ and $\mathsf{Prove}(ck^*, \mathrm{E}, (M'_1, \rho'_1), \ldots, (M'_n, \rho'_n))$ are equally distributed.
- *Randomizability.* Given ck, commitments $\mathbf{c}_1, \ldots, \mathbf{c}_n$, a proof π for $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$ and E, and $\rho'_1, \ldots, \rho'_n \in \mathcal{R}$, algorithm $\mathsf{RdProof}$ outputs a proof π' for the randomized commitments $\mathbf{c}'_i := \mathsf{RdCom}(ck, \mathbf{c}_i, \rho'_i)$.
  For all i, let M_i be the value committed in $\mathbf{c}_i$ and let ρ_i be such that $\mathbf{c}_i = \mathsf{Com}(ck, M_i, \rho_i)$. Then we require that the output of $\mathsf{RdProof}\big(ck, \mathrm{E}, (\mathbf{c}_1, \rho'_1), \ldots, (\mathbf{c}_n, \rho'_n), \pi\big)$ is distributed as the output of $\mathsf{Prove}\big(ck, \mathrm{E}, (M_1, \rho_1 + \rho'_1), \ldots, (M_n, \rho_n + \rho'_n)\big)$. Thus, if $\rho'_1, \ldots, \rho'_n$ are chosen uniformly, then π' and $(\mathbf{c}'_i)_{i=1}^n$ are distributed as $\mathsf{Prove}\big(ck, \mathrm{E}, (M_1, \hat{\rho}_1), \ldots, (M_n, \hat{\rho}_n)\big)$ and $\big(\mathsf{Com}(ck, M_i, \hat{\rho}_i)\big)_{i=1}^n$ with $\hat{\rho}_i$ chosen uniformly from $\mathcal{R}$.

If E is the conjunction of $\mathrm{E}_1, \ldots, \mathrm{E}_k$ over variables $M_1, \ldots, M_n$ (which can be common to several equations) then for $M = (M_1, \ldots, M_n)$ and $\rho = (\rho_1, \ldots, \rho_n)$ we define

$$\mathsf{Prove}(ck, \mathrm{E}, (M, \rho)) := \big(\mathsf{Prove}(ck, \mathrm{E}_j, (M_i, \rho_i)_{i=1}^n)\big)_{j=1}^k$$
$$\mathsf{Ver}(ck, \mathrm{E}, (\mathbf{c}_i)_{i=1}^n, \pi) := \textstyle\bigwedge_{j=1}^k \mathsf{Ver}(ck, \mathrm{E}_j, (\mathbf{c}_i)_{i=1}^n, \pi_j)$$

for $\pi = (\pi_1, \ldots, \pi_k)$.

2.2.3 Digital Signatures

A digital signature scheme **Sig** consists of the following algorithms: $\mathsf{Setup}_\mathrm{S}$ takes as input the security parameter 1^λ and outputs public parameters pp, which define a message space $\mathcal{M}$. On input pp, $\mathsf{KeyGen}_\mathrm{S}$ outputs a pair (vk, sk) of verification and signing key. For $M \in \mathcal{M}$, $\mathsf{Sign}(sk, M)$ outputs a signature Σ, which is verified by $\mathsf{Ver}(vk, M, \Sigma)$ outputting a decision bit. Correctness requires that if pp is output by $\mathsf{Setup}_\mathrm{S}$ and (vk, sk) is output by $\mathsf{KeyGen}_\mathrm{S}(pp)$ then for all $M \in \mathcal{M}$: $\mathsf{Ver}(vk, M, \mathsf{Sign}(sk, M)) = 1$.

Signatures are *existentially unforgeable under chosen-message attack* (EUF-CMA) [GMR88] if no adversary, given vk and a signing oracle for messages of its choice, can output a pair (M, Σ) s.t. M was never queried and $\mathsf{Ver}(vk, M, \Sigma) = 1$. We define two more notions for **Sig**:

- *Strong unforgeability (under chosen message attack).* No probabilistic polynomial-time (p.p.t.) adversary, given vk and an oracle for adaptive signing queries on messages of its choice can output a pair (M, Σ), s.t. $\mathsf{Ver}(vk, M, \Sigma) = 1$ and $(M, \Sigma) \neq (M_i, \Sigma_i)$ for all i; where M_i are the queried messages and Σ_i the oracle responses.
- *Compatibility* with **Com** and **Proof**. The messages, verification keys and signatures consist of values in $\mathcal{V}$, the value space of **Com**. Moreover, the signature verification predicate is a conjunction of equations from $\mathcal{E}$, the class of equations for **Proof**. We denote the verification equations over the variables vk, M and Σ by $\mathrm{E}_{\mathsf{Ver}}(vk, M, \Sigma)$.[3]

[3]The parameters of the signature scheme, which determine the verification equations, will be assumed implicitly. Later we will consider some of the variables as constants. For example, the verification equations over variables M and Σ considering vk as a constant will be denoted as $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}(M, \Sigma)$.

2.2.4 Blind Signatures

A blind signature scheme consists of the algorithms $\mathsf{Setup}_{\mathrm{S}}, \mathsf{KeyGen}_{\mathrm{S}}$ and Ver, as defined for digital signatures, and moreover the interactive protocol $\mathsf{Issue} \leftrightarrow \mathsf{Obtain}$ between the signer and a user allowing the latter to obtain a signature on a message hidden from the signer. Issue takes as input a signing key sk and Obtain has input the corresponding verification key vk and a message M. If they do not abort then the output of Obtain is a signature Σ. Correctness requires that if pp is output by $\mathsf{Setup}_{\mathrm{S}}$ and (vk, sk) is output by $\mathsf{KeyGen}_{\mathrm{S}}(pp)$ then for all $M \in \mathcal{M}$, $\mathsf{Issue}(sk) \leftrightarrow \mathsf{Obtain}(vk, M)$ does not abort and the output Σ satisfies $\mathsf{Ver}(vk, M, \Sigma) = 1$.

(Concurrent) security is defined by the following requirements [HKKL07]:

- *Blindness.* For any p.p.t. algorithm Issue^* the probability of winning the following game is negligibly close to $\frac{1}{2}$: Run $pp \leftarrow \mathsf{Setup}_{\mathrm{S}}$ and give pp to Issue^*, which outputs a key vk and two messages $M_0, M_1 \in \mathcal{M}$. Issue^* interacts concurrently with $\mathsf{Obtain}(vk, M_b)$ and $\mathsf{Obtain}(vk, M_{1-b})$, where the bit b is chosen randomly. If either instance of Obtain aborts, define $\Sigma_0 = \Sigma_1 = \bot$, otherwise let Σ_0 and Σ_1 be the respective outputs of $\mathsf{Obtain}(vk, M_b)$ and $\mathsf{Obtain}(vk, M_{1-b})$. Issue^* is given (Σ_0, Σ_1) and wins if it outputs the bit b.

- *Unforgeability.* For any p.p.t. algorithm Obtain^* and for any polynomial q the probability of winning the following game is negligible: Run $pp \leftarrow \mathsf{Setup}(1^\lambda)$ and $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}(pp)$ and give vk to Obtain^*. Let Obtain^* concurrently interact with $q = q(\lambda)$ instances of $\mathsf{Sign}(sk)$. Obtain^* wins if it outputs $(M_1, \Sigma_1, \ldots, M_{q+1}, \Sigma_{q+1})$, with $M_i \in \mathcal{M}$, $M_i \neq M_j$ and $\mathsf{Ver}(vk, M_i, \Sigma_i) = 1$ for all i, j with $i \neq j$.

2.3 Bilinear Groups

A *bilinear group* is a tuple $grp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ where $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are cyclic groups of prime order p, G_1 and G_2 generate $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient non-degenerate bilinear map, i.e.,

$$\forall\, X \in \mathbb{G}_1\ \forall\, Y \in \mathbb{G}_2\ \forall\, a, b \in \mathbb{Z} : e(X^a, Y^b) = e(X, Y)^{ab} \ ,$$

and $e(G_1, G_2)$ generates $\mathbb{G}_T$. We assume that there exists a probabilistic polynomial-time algorithm GrpGen that on input 1^λ outputs a bilinear group grp for which p is a λ-bit prime.

Bilinear groups are typically instantiated by elliptic curves on which a *pairing* can be defined as a bilinear map. Galbraith et al. [GPS08] distinguish 3 types of pairings: For Type 1 we have $\mathbb{G}_1 = \mathbb{G}_2$ and $G_1 = G_2$; such groups are called *symmetric* and we will write $(p, \mathbb{G}, \mathbb{G}_T, e, G)$. If $\mathbb{G}_1 \neq \mathbb{G}_2$ and there exists an efficiently computable homomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, we say grp is of Type 2. Finally, in Type 3 curves we have $\mathbb{G}_1 \neq \mathbb{G}_2$ and there are no efficiently computable homomorphisms between $\mathbb{G}_1$ and $\mathbb{G}_2$. In such groups it is reasonable to make the SXDH assumption introduced below (whereas it is wrong in groups of Type 1 and 2). Our new assumptions in Chapter 3 are reasonable for all types.

Assumption 1 (SXDH)**.** *The* Symmetric External Diffie-Hellman assumption *states that given* (G_1^r, G_1^s, G_1^t) *for random* $r, s \in \mathbb{Z}_p$*, it is hard to decide whether* $t = rs$ *or* t *is random; likewise, given* $(G_2^{r'}, G_2^{s'}, G_2^{t'})$ *for random* $r', s' \in \mathbb{Z}_p$*, it is hard to decide whether* $t' = r's'$ *or* t' *is random.*

Assumption 2 (DLIN)**.** *The* Decision Linear assumption*, introduced in* [BBS04]*, in a symmetric group* $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ *states that given* $(G^\alpha, G^\beta, G^{r\alpha}, G^{s\beta}, G^t)$ *for random* $\alpha, \beta, r, s \in \mathbb{Z}_p$*, it is hard to decide whether* $t = r + s$ *or* t *is random.*

Assumption 3 (SDH)**.** *The* Strong Diffie-Hellman assumption [BB04] *in an asymmetric group* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ *states that given* $(G_1, G_1^x, G_1^{(x^2)}, \ldots, G_1^{(x^q)}, G_2, G_2^x)$ *for a random* $x \in \mathbb{Z}_p$*, it is hard to produce a pair* $(G_1^{\frac{1}{x+c}}, c) \in \mathbb{G}_1 \times \mathbb{Z}_p$*.*

Assumption 4 (HSDH)**.** *The* Hidden Strong Diffie-Hellman assumption [BW07] *in a symmetric group* $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ *states that given* G, H, G^x *and* $q-1$ *triples* $(G^{\frac{1}{x+c_i}}, G^{c_i}, H^{c_i})$ *for random* $H \in \mathbb{G}$ *and* $c_i \in \mathbb{Z}_p$*, it is hard to produce a new triple* $(G^{\frac{1}{x+c^*}}, G^{c^*}, H^{c^*})$ *with* $c^* \neq c_i$ *for all* i*.*

Throughout the paper, we will assume two fixed generators G and H of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively (with $G \neq H$ when $\mathbb{G}_1 = \mathbb{G}_2$). We call a pair $(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$ a *Diffie-Hellman pair* (w.r.t. (G, H)), if there exists $a \in \mathbb{Z}_p$ such that $A = G^a$ and $B = H^a$. Using the bilinear map e, such pairs are efficiently decidable by checking $e(A, H) = e(G, B)$. We let $\mathcal{DH}$ denote the set of DH pairs and implicitly assume them to be w.r.t. G and H.

2.4 Groth-Sahai Proofs for Pairing-Product Equations

In this section we show how to instantiate **Com** and **Proof** from Sections 2.2.1 and 2.2.2. A compatible signature scheme **Sig** and a blind signature based on it will be introduced in Chapter 4.

We start by presenting a perfectly binding extractable commitment scheme $\mathbf{Com}_{\mathrm{L}}$, which is computationally hiding under DLIN, and then give an overview of Groth-Sahai proofs introduced in [GS08]. This is sufficient if they are to be used in a black-box way. In Sections 2.4.2 and 2.4.3 we give an instantiation of **Com** and **Proof** over asymmetric groups whose security is based on the SXDH assumption. As we will prove new properties of these proofs in Sect. 5.3, we will give the SXDH instantiation in detail.

2.4.1 DLIN Commitments and Proofs

Linear Commitments. For $\mathbf{Com}_{\mathrm{L}}$, let $grp = (p, \mathbb{G}, \mathbb{G}_T, e, G)$ be a symmetric bilinear group in which DLIN holds. $\mathsf{Setup}_{\mathrm{L}}(grp)$ chooses $\alpha, \beta, t_1, t_2 \leftarrow \mathbb{Z}_p$ and outputs

$$ck := \big(\mathbf{u}_1 = (G^\alpha, 1, G), \mathbf{u}_2 = (1, G^\beta, G), \mathbf{u}_3 = (G^{t_1\alpha}, G^{t_2\beta}, G^{t_1+t_2})\big) \ .$$

$\mathsf{ExSetup}_{\mathrm{L}}$ additionally outputs the extraction key $ek := (\alpha, \beta)$. A commitment to $X \in \mathbb{G}$ with randomness $r = (r_1, r_2, r_3) \in \mathcal{R}_{\mathrm{L}} := \mathbb{Z}_p^3$ is defined as

$$\begin{aligned}\mathsf{Com}_{\mathrm{L}}(ck, X, r) &:= \big(\textstyle\prod u_{i1}^{r_i},\ \prod u_{i2}^{r_i},\ X \cdot \prod u_{i3}^{r_i}\big) \\ &= \big(u_{11}^{r_1} \cdot u_{31}^{r_3},\ u_{22}^{r_2} \cdot u_{32}^{r_3},\ X \cdot G^{r_1+r_2} \cdot u_{33}^{r_3}\big) \\ &= \big(G^{\alpha(r_1+t_1r_3)},\ G^{\beta(r_2+t_2r_3)},\ X \cdot G^{r_1+r_2+t_1r_3+t_2r_3}\big) \ ,\end{aligned}$$

which is a *linear encryption* [BBS04] under encryption key (G^α, G^β) using randomness $(r_1 + t_1r_3,\ r_2+t_2r_3)$. $\mathsf{Extr}_{\mathrm{L}}\big((\alpha, \beta), (c_1, c_2, c_3)\big)$ decrypts the ciphertext by outputting $c_3 \cdot c_1^{-1/\alpha} \cdot c_2^{-1/\beta}$. $\mathsf{WISetup}_{\mathrm{L}}$

replaces u_{33} in ck with $G^{t_1+t_2-1}$, which is indistinguishable by DLIN and which leads to perfectly hiding commitments.

Groth-Sahai WI Proofs. Groth and Sahai introduce witness-indistinguishable proofs of *satisfiability of pairing-product equations*. A *pairing-product equation* (PPE) in a symmetric bilinear group over variables $Y_1, \ldots, Y_n \in \mathbb{G}$ is an equation E of the form[4]

$$\mathrm{E}(Y_1, \ldots, Y_n) \ : \ \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{n}\prod_{j=1}^{n} e(\underline{Y_i}, \underline{Y_j})^{\gamma_{i,j}} \ = \ \mathrm{t}_T \ ,$$

determined by $A_i \in \mathbb{G}$, $\gamma_{i,j} \in \mathbb{Z}_p$, for $1 \le i, j \le n$, and $\mathrm{t}_T \in \mathbb{G}_T$. *Witness indistinguishability* (WI) means that proofs computed with different witnesses $(Y_1, \ldots, Y_n)$ are indistinguishable.

Let $\mathbf{Com}_{\mathrm{L}} = (\mathsf{Setup}_{\mathrm{L}}, \mathsf{Com}_{\mathrm{L}}, \mathsf{ExSetup}_{\mathrm{L}}, \mathsf{Extr}_{\mathrm{L}}, \mathsf{WISetup}_{\mathrm{L}})$ denote the commitment scheme from above. The proof system for a symmetric bilinear group grp is set up by running $\mathsf{Setup}_{\mathrm{L}}(grp)$ which produces a perfectly binding commitment key ck. To make a WI proof of satisfiability of an equation, given a witness $(Y_1, \ldots, Y_n) \in \mathbb{G}^n$ satisfying E, one first *commits* to the witness. For every i one chooses randomness $r_i \leftarrow \mathcal{R}_{\mathrm{L}}$ and sets $\mathbf{c}_i := \mathsf{Com}_{\mathrm{L}}(ck, Y_i, r_i)$. Running $\mathsf{Prove}_{\mathrm{L}}(ck, \mathrm{E}, (Y_i, r_i)_{i=1}^n)$ generates a proof[5] π which asserts that the values committed in $\mathbf{c}_i$ satisfy E. In the DLIN instantiation, the proof is in $\mathbb{G}^9$; however, if E is a *linear equation* (i.e., $\gamma_{i,j} = 0$ for all i, j), then the proof reduces to 3 group elements. A proof π for equation E and commitments $(\mathbf{c}_i)_{i=1}^n$ under ck is verified by $\mathsf{Verify}_{\mathrm{L}}(ck, \mathrm{E}, (\mathbf{c}_i)_{i=1}^n, \pi)$. An honestly computed proof for commitments to values satisfying E is always accepted by $\mathsf{Verify}_{\mathrm{L}}$ (completeness).

Security. *Soundness.* Given commitments $(\mathbf{c}_i)_{i=1}^n$ s.t. $\mathsf{Verify}_{\mathrm{L}}(ck, \mathrm{E}, (\mathbf{c}_i)_{i=1}^n, \pi) = 1$ for some π and the extraction key ek output by $\mathsf{ExSetup}_{\mathrm{L}}$, algorithm $\mathsf{Extr}_{\mathrm{L}}$ applied to $\mathbf{c}_i$ for all i yields a vector $(Y_i)_{i=1}^n$ satisfying E.

Witness Indistinguishability. If the commitment key ck output by $\mathsf{Setup}_{\mathrm{L}}$ is replaced by ck^* output by $\mathsf{WISetup}_{\mathrm{L}}$ (which is indistinguishable under DLIN) then $\mathsf{Com}_{\mathrm{L}}$ commitments are perfectly hiding; i.e., given $\mathbf{c} \in \mathbb{G}^3$, then for any $Y \in \mathbb{G}$ there exists an $r \in \mathcal{R}_{\mathrm{L}}$ s.t. $\mathbf{c} = \mathsf{Com}_{\mathrm{L}}(ck^*, Y, r)$. Given values $((Y_1, r_1), \ldots, (Y_n, r_n))$ and $((Y_1', r_1'), \ldots, (Y_n', r_n'))$ such that for all $i = 1, \ldots, n$ we have $\mathsf{Com}_{\mathrm{L}}(ck^*, Y_i, r_i) = \mathsf{Com}_{\mathrm{L}}(ck^*, Y_i', r_i')$, and moreover $(Y_1, \ldots, Y_n)$ and $(Y_1', \ldots, Y_n')$ both satisfy E, then $\mathsf{Prove}_{\mathrm{L}}(ck^*, \mathrm{E}, (Y_i, r_i)_{i=1}^n)$ and $\mathsf{Prove}_{\mathrm{L}}(ck^*, \mathrm{E}, (Y_i', r_i')_{i=1}^n)$ generate the same distribution of proofs. Thus in the WI setting (i.e., when ck is computed by $\mathsf{WISetup}_{\mathrm{L}}$) the commitments and the proof hide the values in an information-theoretical sense.

Randomizing Groth-Sahai Proofs. As observed in [3] and [BCC+09] and formalized by the latter, Groth-Sahai WI proofs of knowledge can be *randomized*. This means that there exists an algorithm $\mathsf{RdCom}_{\mathrm{L}}$ that on input ck, a commitment $\mathbf{c}$ and fresh randomness $r' \in \mathcal{R}_{\mathrm{L}}$ outputs a *randomization* of $\mathbf{c}$ under r', which is a commitment $\mathbf{c}'$ to the same value but under a different randomness.

A proof π for an equation E and a vector of commitments $(\mathbf{c}_i)_{i=1}^n$ can be *adapted* (and randomized itself) to the randomizations $\mathbf{c}_i' = \mathsf{RdCom}_{\mathrm{L}}(ck, \mathbf{c}_i, r_i')$ without knowledge of the committed values: running $\mathsf{RdProof}_{\mathrm{L}}(ck, \mathrm{E}, (\mathbf{c}_i, r_i')_{i=1}^n, \pi)$ computes π' for equation E and commitments $\mathbf{c}_i'$. If the r_i' are chosen uniformly from $\mathcal{R}_{\mathrm{L}}$ then $((\mathbf{c}_i')_{i=1}^n, \pi')$ is distributed as

$$\Big[(\hat{r}_1, \ldots, \hat{r}_n) \leftarrow \mathcal{R}_{\mathrm{L}}^n \ : \ \big(\big(\mathsf{Com}_{\mathrm{L}}(ck, Y_i, \hat{r}_i)\big)_{i=1}^n, \ \mathsf{Prove}_{\mathrm{L}}\big(ck, \mathrm{E}, (Y_i, \hat{r}_i)_{i=1}^n\big)\big)\Big]$$

[4] For a more concise exposition we will underline the variables of an equation.

[5] Note that in this context the word *proof* can either denominate "proof of satisfiability" (or language-membership)—which thus includes the commitments—or mean a proof *that the content of some given commitments satisfies a given equation*. We adopt the latter diction, and say *proof of knowledge* when we include the commitments.

(and therefore by completeness: $\mathsf{Verify}_{\mathrm{L}}(ck, \mathrm{E}, (\mathbf{c}'_i)_{i=1}^n, \pi') = 1$). Basically, if for all i, r_i is the randomness of the original commitments then $\mathbf{c}'_i = \mathsf{Com}_{\mathrm{L}}(ck, Y_i, r_i + r'_i)$, and π' is distributed as proofs output by $\mathsf{Prove}_{\mathrm{L}}(ck, \mathrm{E}, (Y_i, r_i + r'_i)_{i=1}^n)$. (This is shown analogously to Remark 1 in Sect. 2.4.3 below.)

Examples. (1) *Proof of two commitments containing the same value.* Let $\mathrm{E}_{\mathrm{equal}}(X_1, X_2)$ denote the equation $e(\underline{X_1}, H)\, e(\underline{X_2}, H^{-1}) = 1$. Given two commitments $\mathbf{c} = \mathsf{Com}(ck, M, r)$ and $\mathbf{c}' = \mathsf{Com}(ck, M', r')$, $\mathsf{Prove}(ck, \mathrm{E}_{\mathrm{equal}}, (M, r), (M', r'))$ proves that $\mathbf{c}$ and $\mathbf{c}'$ commit to the same value.

(2) *Proof of commitments to a DH-pair.* Define $\mathrm{E}_{\mathcal{DH}}(X, Y)$ as $e(\underline{X}, H)\, e(G^{-1}, \underline{Y}) = 1$. A proof for Equation $\mathrm{E}_{\mathcal{DH}}$ and commitments $\mathbf{c}_X$ and $\mathbf{d}_Y$ proves that the pair of committed values is in $\mathcal{DH}$. Under $\mathbf{Com}_{\mathrm{L}}$, the proof is in $\mathbb{G}^3$.

2.4.2 SXDH Commitments

We instantiate **Com**, defined in Sect. 2.2.1, by the commitment scheme based on SXDH given in [GS08].

Setup on input an asymmetric group $grp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ chooses $\alpha_1, \alpha_2, t_1, t_2 \leftarrow \mathbb{Z}_p$ and returns $ck = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1, \mathbf{v}_2)$ with

$$\mathbf{u}_1 := (G_1, G_1^{\alpha_1}) \qquad \mathbf{u}_2 := (G_1^{t_1}, G_1^{\alpha_1 t_1}) \qquad \mathbf{v}_1 := (G_2, G_2^{\alpha_2}) \qquad \mathbf{v}_2 := (G_2^{t_2}, G_2^{\alpha_2 t_2}) \tag{2.1}$$

Value and random space are defined as $\mathcal{V} := \mathbb{G}_1 \cup \mathbb{G}_2$ and $\mathcal{R} := \mathbb{Z}_p^2$.

$\mathsf{Com}(ck, X, r)$ is defined as follows: for $X \in \mathbb{G}_1$, $\mathsf{Com}(ck, X, r) := (u_{11}^{r_1} \cdot u_{21}^{r_2},\ X \cdot u_{12}^{r_1} \cdot u_{22}^{r_2})$; for $X \in \mathbb{G}_2$, $\mathsf{Com}(ck, X, r) := (v_{11}^{r_1} \cdot v_{21}^{r_2},\ X \cdot v_{12}^{r_1} \cdot v_{22}^{r_2})$.

$\mathsf{RdCom}(ck, \mathbf{c}, r')$ returns $\mathbf{c} \circ \mathsf{Com}(ck, 1, r') = (c_1 \cdot u_{11}^{r'_1} \cdot u_{21}^{r'_2},\ c_2 \cdot u_{12}^{r'_1} \cdot u_{22}^{r'_2})$, when $\mathbf{c} \in \mathbb{G}_1^2$ and similarly for the case when $\mathbf{c} \in \mathbb{G}_2^2$. ("$\circ$" denotes component-wise multiplication.)

$\mathsf{ExSetup}$ constructs ck as in Setup and in addition outputs the extraction key $ek := (\alpha_1, \alpha_2)$.

$\mathsf{Extr}(ek, \mathbf{c})$ is defined as follows. If $\mathbf{c} \in \mathbb{G}_1^2$ then output $c_2 \cdot c_1^{-\alpha_1}$; if $\mathbf{c} \in \mathbb{G}_2^2$ then output $c_2 \cdot c_1^{-\alpha_2}$.

$\mathsf{WISetup}$ produces ck as Setup, but sets u_{22} and v_{22} as $G_1^{\alpha_1 t_1 - 1}$ and $G_2^{\alpha_2 t_2 - 1}$, respectively. This is indistinguishable by SXDH and results in perfectly hiding commitments.

Security. *The scheme is perfectly binding, computationally hiding and randomizable as defined in Sect. 2.2.1.*

Let i be 1 or 2. For $X \in \mathbb{G}_i$ we have $\mathsf{Com}(ck, X, r) = (G_i^{r_1} \cdot G_i^{t_i r_2}, X \cdot G_i^{\alpha_i r_1} \cdot G_i^{\alpha_i t_i r_2}) = (G_i^{r_1 + t_i r_2}, X \cdot (G_i^{\alpha_i})^{r_1 + t_i r_2})$, which is an ElGamal encryption under public key $G_i^{\alpha_i}$ and randomness $r_1 + t_i r_2$. $\mathsf{Extr}(ek, \mathbf{c})$ decrypts thus $\mathbf{c} = (c_1, c_2)$ by outputting $c_2 \cdot c_1^{-\alpha_i}$. Moreover, **Com** commitments are *homomorphic*: $\mathsf{Com}(ck, X, r) \circ \mathsf{Com}(ck, X', r') = \mathsf{Com}(ck, X \cdot X', r + r')$; therefore if $\mathbf{c} = \mathsf{Com}(ck, X, r)$ then $\mathsf{RdCom}(ck, \mathbf{c}, r') = \mathsf{Com}(ck, X, r + r')$.

For ck^* output by $\mathsf{WISetup}$ we have $\mathsf{Com}(ck^*, X, r) = (G_i^{r_1 + t_i r_2}, X \cdot G_i^{\alpha_i r_1 + \alpha_i t_i r_2 - r_2})$, which is uniformly random in $\mathbb{G}_i^2$ for $(r_1, r_2) \leftarrow \mathbb{Z}_p$.

2.4.3 SXDH Groth-Sahai Proofs for Pairing-Product Equations

In order to instantiate **Proof**, defined in Sect. 2.2.2, we use the proof system introduced in [GS08] and shown to be randomizable in [BCC+09]. The class of equations $\mathcal{E}$ for our proof system are *pairing-product equations* (PPE). A PPE over variables $X_1, \ldots, X_m \in \mathbb{G}_1$ and $Y_1, \ldots, Y_n \in \mathbb{G}_2$ is an equation of the form

$$\mathrm{E}(X_1, \ldots, X_m; Y_1, \ldots, Y_n) \;:\; \prod_{j=1}^{n} e(A_j, \underline{Y_j}) \prod_{j=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \;=\; \mathfrak{t}_T \;, \tag{2.2}$$

defined by $A_j \in \mathbb{G}_1$, $B_i \in \mathbb{G}_2$, $\gamma_{i,j} \in \mathbb{Z}_p$, for $1 \le i \le m$ and $1 \le j \le n$, and $\mathfrak{t}_T \in \mathbb{G}_T$.

Proofs. We define $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n\big)$ for $X_i \in \mathbb{G}_1, Y_j \in \mathbb{G}_2$ and $r_i, s_j \in \mathbb{Z}_p^2$, and equation E given by the values $\big((A_j)_{j=1}^n, (B_i)_{i=1}^m, (\gamma_{i,j})_{i,j} \in \mathbb{Z}_p^{m\times n}, \mathfrak{t}_T \in \mathbb{G}_T\big)$. For notational convenience, let us first define the following two shortcuts[6] for $Z = (z_{ij})_{ij} \in \mathbb{Z}_p^{2\times 2}, \vec{\mathbf{u}} \in \mathbb{G}_1^{2\times 2}, \vec{\mathbf{v}} \in \mathbb{G}_2^{2\times 2}$.

$$Z \otimes \vec{\mathbf{u}} := \begin{bmatrix} u_{11}^{z_{11}} u_{21}^{z_{12}} & u_{12}^{z_{11}} u_{22}^{z_{12}} \\ u_{11}^{z_{21}} u_{21}^{z_{22}} & u_{12}^{z_{21}} u_{22}^{z_{22}} \end{bmatrix} \qquad Z \otimes \vec{\mathbf{v}} := \begin{bmatrix} v_{11}^{-z_{11}} v_{21}^{-z_{21}} & v_{12}^{-z_{11}} v_{22}^{-z_{21}} \\ v_{11}^{-z_{12}} v_{21}^{-z_{22}} & v_{12}^{-z_{12}} v_{22}^{-z_{22}} \end{bmatrix} \tag{2.3}$$

Prove chooses $Z = ((z_{11}, z_{12}), (z_{21}, z_{22}))^\top \leftarrow \mathbb{Z}_p^{2\times 2}$ and defines

$$\begin{aligned} t_{11} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij} s_{j1} & t_{12} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij} s_{j2} \\ t_{21} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij} s_{j1} & t_{22} &:= \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij} s_{j2} \end{aligned} \tag{2.4}$$

The output $\pi = (\phi, \theta) \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ of Prove is then defined as:

$$\begin{aligned} \phi &:= \begin{bmatrix} v_{11}^{t_{11}} v_{21}^{t_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i1}\gamma_{ij}}\big) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ v_{11}^{t_{21}} v_{21}^{t_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \circ (Z \otimes \vec{\mathbf{v}}) \\ \theta &:= \begin{bmatrix} 1 & \big(\prod_{j=1}^n A_j^{s_{j1}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j1}\gamma_{ij}}\big) \\ 1 & \big(\prod_{j=1}^n A_j^{s_{j2}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j2}\gamma_{ij}}\big) \end{bmatrix} \circ (Z \otimes \vec{\mathbf{u}}) \end{aligned} \tag{2.5}$$

We write $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n; Z\big)$ if we want to make the internal randomness $Z \in \mathbb{Z}_p^{2\times 2}$ explicit.

If for E we have $\gamma_{ij} = 0$ and $A_j = 1$ (or $B_i = 1$) for all i, j then E is called *linear*. In this case we can choose $Z := 0$, which makes all but 2 elements of (ϕ, θ) equal to 1. The proof is thus in $\mathbb{G}_2^2$ (or in $\mathbb{G}_1^2$ if all $B_i = 1$).

Randomization. Randomization of a commitment $\mathbf{c} = \mathsf{Com}(ck, X, r)$ via $\mathsf{RdCom}(ck, \mathbf{c}, r')$ replaces randomness r by randomness $r + r'$; and similarly for $\mathbf{d} = \mathsf{Com}(ck, Y, s)$. Adaptation of a proof by $\mathsf{RdProof}$ must thus replace the values r_i and s_j in $\pi = (\phi, \theta)$ analogously. We formally define:

$\mathsf{RdProof}\big(ck, \mathrm{E}, (\mathbf{c}_i, r_i)_{i=1}^m, (\mathbf{d}_j, s_j)_{j=1}^n, \pi\big)$: choose $Z = ((z_{11}, z_{12}), (z_{21}, z_{22}))^\top \leftarrow \mathbb{Z}_p^{2\times 2}$ and define $(t_{11}, t_{12}, t_{21}, t_{22})$ as in (2.4). Output $\pi' = (\phi', \theta') \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ defined as:

[6] If we denoted $\mathbb{G}_1$ and $\mathbb{G}_2$ additively (as was done in [GS08]), the shortcuts would correspond to matrix multiplication: $Z \otimes \vec{\mathbf{u}} = Z\vec{\mathbf{u}}$ and $Z \otimes \vec{\mathbf{v}} = -Z^\top \vec{\mathbf{v}}$.

$$\phi' := \phi \circ \begin{bmatrix} \left(\prod_{j=1}^{n} d_{j1}^{\sum_{i=1}^{m} r_{i1}\gamma_{ij}}\right) v_{11}^{t_{11}} v_{21}^{t_{12}} & \left(\prod_{i=1}^{m} B_i^{r_{i1}}\right)\left(\prod_{j=1}^{n} d_{j2}^{\sum_{i=1}^{m} r_{i1}\gamma_{ij}}\right) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ \left(\prod_{j=1}^{n} d_{j1}^{\sum_{i=1}^{m} r_{i2}\gamma_{ij}}\right) v_{11}^{t_{21}} v_{21}^{t_{22}} & \left(\prod_{i=1}^{m} B_i^{r_{i2}}\right)\left(\prod_{j=1}^{n} d_{j2}^{\sum_{i=1}^{m} r_{i2}\gamma_{ij}}\right) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \circ (Z \otimes \vec{\mathbf{v}})$$

$$\theta' := \theta \circ \begin{bmatrix} \left(\prod_{i=1}^{m} c_{i1}^{\sum_{j=1}^{n} s_{j1}\gamma_{ij}}\right) & \left(\prod_{j=1}^{n} A_j^{s_{j1}}\right)\left(\prod_{i=1}^{m} c_{i2}^{\sum_{j=1}^{n} s_{j1}\gamma_{ij}}\right) \\ \left(\prod_{i=1}^{m} c_{i1}^{\sum_{j=1}^{n} s_{j2}\gamma_{ij}}\right) & \left(\prod_{j=1}^{n} A_j^{s_{j2}}\right)\left(\prod_{i=1}^{m} c_{i2}^{\sum_{j=1}^{n} s_{j2}\gamma_{ij}}\right) \end{bmatrix} \circ (Z \otimes \vec{\mathbf{u}})$$

which has the same distribution as the output of $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r'_i + r_i)_{i=1}^m, (Y_j, s'_j + s_j)_{j=1}^n\big)$, where r'_i and s'_j are such that $\mathbf{c}_i = \mathsf{Com}(ck, X_i, r'_i)$ and $\mathbf{d}_j = \mathsf{Com}(ck, Y_j, s'_j)$, as we will show now.

Remark 1. In additive notation, the commitments and proofs can be written as follows (cf. the full version of [GS08]), when the cumulated randomness for all variables is $R = (r_{ik}) \in \mathbb{Z}_p^{m\times 2}$ and $S = (s_{jk}) \in \mathbb{Z}_p^{n\times 2}$, and we set $\Gamma = (\gamma_{ij}) \in \mathbb{Z}_p^{m\times n}$ and define $\iota_1(\vec{X}) := [\,\vec{0}\,|\,\vec{X}\,]$ and $\iota_2(\vec{Y}) := [\,\vec{0}\,|\,\vec{Y}\,]$, for $\vec{X} \in \mathbb{G}_1^m$ and $\vec{Y} \in \mathbb{G}_2^n$.

$$\vec{\mathbf{c}} = \iota_1(\vec{X}) + R\vec{\mathbf{u}} \qquad \phi = R^\top \iota_2(\vec{B}) + R^\top \Gamma \iota_2(\vec{Y}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}$$
$$\vec{\mathbf{d}} = \iota_2(\vec{Y}) + S\vec{\mathbf{v}} \qquad \theta = S^\top \iota_1(\vec{A}) + S^\top \Gamma^\top \iota_1(\vec{X}) + Z\vec{\mathbf{u}}$$

To randomize the commitments and proofs, choose $\hat{R} \leftarrow \mathbb{Z}_p^{m\times 2}$, $\hat{S} \leftarrow \mathbb{Z}_p^{n\times 2}$, $\hat{Z} \leftarrow \mathbb{Z}_p^{2\times 2}$ and set

$$\begin{aligned}
\vec{\mathbf{c}}' &:= \vec{\mathbf{c}} + \hat{R}\vec{\mathbf{u}} = \iota_1(\vec{X}) + (R + \hat{R})\vec{\mathbf{u}} \\
\vec{\mathbf{d}}' &:= \vec{\mathbf{d}} + \hat{S}\vec{\mathbf{v}} = \iota_2(\vec{Y}) + (S + \hat{S})\vec{\mathbf{v}} \\
\phi' &:= \phi + \hat{R}^\top \iota_2(\vec{B}) + \hat{R}^\top \Gamma \vec{\mathbf{d}} + (\hat{R}^\top \Gamma \hat{S} - \hat{Z}^\top)\vec{\mathbf{v}} \\
&= \big[R^\top \iota_2(\vec{B}) + R^\top \Gamma \iota_2(\vec{Y}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}\big] + \hat{R}^\top \iota_2(\vec{B}) + \hat{R}^\top \Gamma \big[\iota_2(\vec{Y}) + S\vec{\mathbf{v}}\big] + (\hat{R}^\top \Gamma \hat{S} - \hat{Z}^\top)\vec{\mathbf{v}} \\
&= (R + \hat{R})^\top \iota_2(\vec{B}) + (R + \hat{R})^\top \Gamma \iota_2(\vec{Y}) + \big[(R + \hat{R})^\top \Gamma (S + \hat{S}) - \underbrace{(Z^\top + \hat{Z}^\top + R^\top \Gamma \hat{S})}_{=(Z + \hat{Z} + \hat{S}^\top \Gamma^\top R)^\top =: (Z')^\top}\big]\vec{\mathbf{v}} \\
\theta' &:= \theta + \hat{S}^\top \iota_1(\vec{A}) + \hat{S}^\top \Gamma^\top \vec{\mathbf{c}} + \hat{Z}\vec{\mathbf{u}} \\
&= \big[S^\top \iota_1(\vec{A}) + S^\top \Gamma^\top \iota_1(\vec{X}) + Z\vec{\mathbf{u}}\big] + \hat{S}^\top \iota_1(\vec{A}) + \hat{S}^\top \Gamma^\top \big[\iota_1(\vec{X}) + R\vec{\mathbf{u}}\big] + \hat{Z}\vec{\mathbf{u}} \\
&= (S + \hat{S})^\top \iota_1(\vec{A}) + (S + \hat{S})^\top \Gamma^\top \iota_1(\vec{X}) + \underbrace{(Z + \hat{Z} + \hat{S}^\top \Gamma^\top R)}_{=Z'}\vec{\mathbf{u}}
\end{aligned}$$

The output of $\mathsf{RdProof}\big(ck, \mathrm{E}, (\mathbf{c}_i, \hat{r}_i)_{i=1}^m, (\mathbf{c}_j, \hat{s}_j)_{j=1}^n, \pi\big)$ using randomness $((\hat{z}_{11}, \hat{z}_{12}), (\hat{z}_{21}, \hat{z}_{22}))^\top$ is therefore the same as that of $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i + \hat{r}_i)_{i=1}^m, (Y_j, s_j + \hat{s}_j)\big)$ when the randomness used is

$$\begin{bmatrix} z_{11} + \hat{z}_{11} + \sum\sum \hat{s}_{j1}\gamma_{ij} r_{i1} & z_{12} + \hat{z}_{12} + \sum\sum \hat{s}_{j1}\gamma_{ij} r_{i2} \\ z_{21} + \hat{z}_{21} + \sum\sum \hat{s}_{j2}\gamma_{ij} r_{i1} & z_{22} + \hat{z}_{22} + \sum\sum \hat{s}_{j2}\gamma_{ij} r_{i2} \end{bmatrix}, \tag{2.6}$$

which is uniformly distributed over $\mathbb{Z}_p^{2\times 2}$ if $\hat{Z}$ is.

Verification. Let $ck = (\vec{\mathbf{u}}, \vec{\mathbf{v}}) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$ be a commitment key, let $\vec{\mathbf{c}} \in \mathbb{G}_1^{m\times 2}, \vec{\mathbf{d}} \in \mathbb{G}_2^{n\times 2}$ be vectors of commitments, and let $(\phi, \theta) \in \mathbb{G}_2^{2\times 2} \times \mathbb{G}_1^{2\times 2}$ be a proof for an equation E given by

$\vec{A} \in \mathbb{G}_1^n$, $\vec{B} \in \mathbb{G}_2^m$, $\Gamma = (\gamma_{i,j})_{i,j} \in \mathbb{Z}_p^{m \times n}$, and $\mathfrak{t}_T \in \mathbb{G}_T$. $\mathsf{Verify}(ck, \mathrm{E}, \vec{\mathbf{c}}, \vec{\mathbf{d}}, (\phi, \theta))$ outputs 1 if and only if the following 4 equations hold.

$$\begin{aligned}
\textstyle\prod_{i=1}^m e\big(c_{i1},\, \prod_{j=1}^n d_{j1}^{\gamma_{ij}}\big) &= e(u_{11}, \phi_{11})\, e(u_{21}, \phi_{21})\, e(\theta_{11}, v_{11})\, e(\theta_{21}, v_{21}) \\
\textstyle\prod_{i=1}^m e\big(c_{i1},\, B_i \prod_{j=1}^n d_{j2}^{\gamma_{ij}}\big) &= e(u_{11}, \phi_{12})\, e(u_{21}, \phi_{22})\, e(\theta_{11}, v_{12})\, e(\theta_{21}, v_{22}) \\
\textstyle\prod_{j=1}^n e\big(A_j \prod_{i=1}^m c_{i2}^{\gamma_{ij}},\, d_{j1}\big) &= e(u_{12}, \phi_{11})\, e(u_{22}, \phi_{21})\, e(\theta_{12}, v_{11})\, e(\theta_{22}, v_{21}) \\
\textstyle\prod_{j=1}^n e(A_j, d_{j2}) \prod_{i=1}^m e\big(c_{i2},\, B_i \prod_{j=1}^n d_{j2}^{\gamma_{ij}}\big) &= \mathfrak{t}_T\, e(u_{12}, \phi_{12})\, e(u_{22}, \phi_{22})\, e(\theta_{12}, v_{12})\, e(\theta_{22}, v_{22})
\end{aligned}$$

We will show completeness, soundness and witness indistinguishability of this proof system in the section below.

Groth and Sahai [GS08] show how to turn their witness-indistinguishable proofs into *zero-knowledge* proofs. In [9] we show how to even achieve *simulation-sound* zero knowledge [Sah99], while maintaining efficiency. (Standard) zero knowledge requires that there exist a simulation setup, which outputs a *trapdoor* and a common reference string (CRS) that is indistinguishable from a regular CRS. Moreover, a proof simulator that is given the trapdoor must be able to produce proofs without being given a witness that are indistinguishable from proofs output by Prove. Soundness of the proof system demands that it be hard for an adversary to generate a valid proof of a false statement. The system is *simulation-sound* if this is even hard when the CRS is simulated and when the adversary has access to a proof-simulator oracle for arbitrary (possibly false) statements.

2.4.4 Correctness, Soundness and Witness Indistinguishability

The functionality of the Groth-Sahai proof system is best explained on a more abstract level—and denoting the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ additively, as already done in Remark 1. Analogously to the bilinear map $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ for group elements, we define a bilinear map F for commitments: $F\colon \mathbb{G}_1^2 \times \mathbb{G}_2^2 \to \mathbb{G}_T^4$ with

$$F\big((c_1, c_2), (d_1, d_2)\big) := \begin{bmatrix} e(c_1, d_1) & e(c_1, d_2) \\ e(c_2, d_1) & e(c_2, d_2) \end{bmatrix} .$$

We now extend the two maps e and F to vectors: "$\cdot$" extends e to vectors of group elements and "$\bullet$" extends F to vectors of pairs of group elements.

$$\begin{aligned}
&\cdot\colon \mathbb{G}_1^m \times \mathbb{G}_2^m \to \mathbb{G}_T && \bullet\colon \mathbb{G}_1^{m \times 2} \times \mathbb{G}_2^{m \times 2} \to \mathbb{G}_T^{2 \times 2} \\
&\vec{X} \cdot \vec{Y} := \prod_{i=1}^m e(X_i, Y_i) && \vec{\mathbf{c}} \bullet \vec{\mathbf{d}} := \begin{pmatrix} \prod_{i=1}^m e(c_{i1}, d_{i1}) & \prod_{i=1}^m e(c_{i1}, d_{i2}) \\ \prod_{i=1}^m e(c_{i2}, d_{i1}) & \prod_{i=1}^m e(c_{i2}, d_{i2}) \end{pmatrix}
\end{aligned}$$

In this notation, the pairing product equation in (2.2) can then be written as

$$(\vec{A} \cdot \vec{Y})\, (\vec{X} \cdot \vec{B})\, (\vec{X} \cdot \Gamma \vec{Y}) = \mathfrak{t}_T \ , \tag{2.7}$$

and the verification equations for a proof (ϕ, θ) above can then be rewritten as

$$\big(\iota_1(\vec{A}) \bullet \vec{\mathbf{d}}\big) \circ \big(\vec{\mathbf{c}} \bullet \iota_2(\vec{B})\big) \circ \big(\vec{\mathbf{c}} \bullet \Gamma \vec{\mathbf{d}}\big) = \iota_T(\mathfrak{t}_T) \circ \big(\vec{\mathbf{u}} \bullet \phi\big) \circ \big(\theta \bullet \vec{\mathbf{v}}\big) \ , \tag{2.8}$$

with $\iota_T\colon \mathfrak{t}_T \mapsto \begin{bmatrix} 1 & 1 \\ 1 & \mathfrak{t}_T \end{bmatrix}$. In additive notation the commitment keys from (2.1) are written as

$$\mathbf{u}_1 := (G_1, \alpha_1 G_1) \qquad \mathbf{u}_2 := (t_1 G_1, \alpha_1 t_1 G_1) \qquad \mathbf{v}_1 := (G_2, \alpha_2 G_2) \qquad \mathbf{v}_2 := (t_2 G_2, \alpha_2 t_2 G_2)$$

Using α_1 and α_2 we can define an *inverse* of ι_1 and ι_2. While ι_i is an *injection* from $\mathbb{G}_i$ to $\mathbb{G}_i^2$, κ_i is a *projection* from $\mathbb{G}_i^2$ onto $\mathbb{G}_i$. For $\mathbf{c} \in \mathbb{G}_1^2$ we define $\kappa_1(c_1, c_2) := c_2 - \alpha_1 c_1$ and for $\mathbf{d} \in \mathbb{G}_2^2$: $\kappa_2(d_1, d_2) := d_2 - \alpha_2 d_1$. Analogously to the injections ι_1 and ι_2, we extend the projections κ_1 and κ_2 to vectors of group elements by applying the projection component-wise.

We have $\kappa_i \circ \iota_i$ is the identity map in $\mathbb{G}_i$ and for $ck = (\vec{\mathbf{u}}, \vec{\mathbf{v}}) \leftarrow \mathsf{Setup}$, we have $\kappa_1(\mathbf{u}_1) = \kappa_1(\mathbf{u}_2) = 0$ and $\kappa_2(\mathbf{v}_1) = \kappa_2(\mathbf{v}_2) = 0$ and κ is the inverse of $\mathsf{Com}(ck, \cdot)$—and is defined as Extr. We also define an inverse of ι_T as $\kappa_T \colon \mathbb{G}_T^{2\times 2} \to \mathbb{G}_T$ with $\kappa_T(((\mathfrak{t}_{11}, \mathfrak{t}_{12}), (\mathfrak{t}_{21}, \mathfrak{t}_{22}))^\top) := \mathfrak{t}_{22}\mathfrak{t}_{12}^{-\alpha_1}(\mathfrak{t}_{21}\mathfrak{t}_{11}^{-\alpha_1})^{\alpha_2}$. The key property of the injections, projections and bilinear maps is that they *commute*, since we have:

$$\begin{aligned} \forall X \in \mathbb{G}_1\ \forall Y \in \mathbb{G}_2 : F(\iota_1(X), \iota_2(Y)) &= \iota_T(e(X, Y)) \\ \forall\, \mathbf{c} \in \mathbb{G}_2^2\ \forall\, \mathbf{d} \in \mathbb{G}_2^2 :\ e(\kappa_1(\mathbf{c}), \kappa_2(\mathbf{d})) &= \kappa_T(F(\mathbf{c}, \mathbf{d})) \end{aligned}$$

This immediately yields soundness of Groth-Sahai proofs as follows. If there exist elements ϕ and θ satisfying (2.8) then applying κ_T to the equation yields

$$\left(\vec{A} \cdot \kappa_2(\vec{\mathbf{d}})\right)\left(\kappa_1(\vec{\mathbf{c}}) \cdot \vec{B}\right)\left(\kappa_1(\vec{\mathbf{c}}) \cdot \Gamma\, \kappa_2(\vec{\mathbf{d}})\right) \;=\; \mathfrak{t}_T\left(0 \cdot \kappa_2(\phi)\right)\left(\kappa_1(\theta) \cdot 0\right) \;=\; \mathfrak{t}_T \ ,$$

since $\kappa_T \circ \iota_T$, $\kappa_1 \circ \iota_1$ and $\kappa_2 \circ \iota_2$ are the identity maps and $\kappa_1(\vec{\mathbf{u}})$ and $\kappa_2(\vec{\mathbf{v}})$ are the zero vectors in the respective groups. The vectors of group elements $\kappa_1(\vec{\mathbf{c}})$ and $\kappa_2(\vec{\mathbf{d}})$ thus satisfy (2.7).

Let us now look at how to construct such a proof and how to make it witness indistinguishable. Given vectors of group elements $\vec{X}$ and $\vec{Y}$ satisfying (2.7) and having defined commitments

$$\vec{\mathbf{c}} = \iota_1(\vec{X}) + R\vec{\mathbf{u}} \qquad\qquad \vec{\mathbf{d}} = \iota_2(\vec{Y}) + S\vec{\mathbf{v}}$$

let us look at the left-hand side of the verification equation in (2.8):

$$\left(\iota_1(\vec{A}) \bullet \vec{\mathbf{d}}\right) \circ \left(\vec{\mathbf{c}} \bullet \iota_2(\vec{B})\right) \circ \left(\vec{\mathbf{c}} \bullet \Gamma\vec{\mathbf{d}}\right) \ . \tag{2.9}$$

Plugging in the definitions of $\vec{\mathbf{c}}$ and $\vec{\mathbf{d}}$ and rearranging terms we get

$$\begin{aligned} &\left(\iota_1(\vec{A}) \bullet \iota_2(\vec{Y})\right) \circ \left(\iota_1(\vec{X}) \bullet \iota_2(\vec{B})\right) \circ \left(\iota_1(\vec{X}) \bullet \Gamma\iota_2(\vec{Y})\right) \\ &\qquad \circ \left(\iota_1(\vec{A}) \bullet S\vec{\mathbf{v}}\right) \circ \left(R\vec{\mathbf{u}} \bullet \iota_2(\vec{B})\right) \circ \left(\iota_1(\vec{X}) \bullet \Gamma S\vec{\mathbf{v}}\right) \circ \left(R\vec{\mathbf{u}} \bullet \Gamma\iota_2(\vec{Y})\right) \circ \left(R\vec{\mathbf{u}} \bullet \Gamma S\vec{\mathbf{v}}\right) \ . \end{aligned}$$

Since e and $\bullet$ commute and $\vec{X}$ and $\vec{Y}$ satisfy (2.7), the first line of the above is equal to $\iota_T(\mathfrak{t}_T)$. Moreover, by bilinearity of $\bullet$ we have $\vec{\mathbf{c}} \bullet \Gamma\vec{\mathbf{d}} = \Gamma^\top\vec{\mathbf{c}} \bullet \vec{\mathbf{d}}$ for all $\vec{\mathbf{c}} \in \mathbb{G}_1^{m\times 2}, \vec{\mathbf{d}} \in \mathbb{G}_2^{n\times 2}, \Gamma \in \mathbb{Z}_p^{m\times n}$. Together this implies that the above expression is equal to

$$\iota_T(\mathfrak{t}_T) \circ \Big(\vec{\mathbf{u}} \bullet \big(\underbrace{R^\top\iota_2(\vec{B}) + R^\top\Gamma\iota_2(\vec{Y}) + R^\top\Gamma S\vec{\mathbf{v}}}_{:=\phi'}\big)\Big) \circ \Big(\big(\underbrace{S^\top\iota_1(\vec{A}) + S^\top\Gamma^\top\iota_1(\vec{X})}_{=:\theta'}\big) \bullet \vec{\mathbf{v}}\Big) \ . \tag{2.10}$$

The expressions ϕ' and θ' defined above satisfy the verification relation since (2.9) equals (2.10). However, to achieve witness indistinguishability we have to *randomize* them. When constructing a proof one chooses $Z \leftarrow \mathbb{Z}_p^{2\times 2}$ and sets $\theta := \theta' + Z\mathbf{u}$ and $\phi := \phi' - Z^\top\mathbf{v}$. Since we have

$$\left(\mathbf{u} \bullet (\phi' - Z^\top\mathbf{v})\right) \circ \left((\theta' + Z\mathbf{u}) \bullet \mathbf{v}\right) = (\mathbf{u} \bullet \phi') \circ (\theta' \bullet \mathbf{v}) \ ,$$

the randomized pair still satisfies the verification relation. We ended up with the definition of ϕ and θ as in Remark 1.

We have shown (perfect) soundness and (perfect) completeness, what remains is witness indistinguishability. For $(\mathbf{u}^*, \mathbf{v}^*) \leftarrow \mathsf{WISetup}$, $\mathbf{u}_1^*$ and $\mathbf{u}_2^*$ are linearly independent, as well as $\mathbf{v}_1^*$ and

$\mathbf{v}_2^*$; we have thus $\mathbb{G}_1^2 \subseteq \langle \mathbf{u}_1^*, \mathbf{u}_2^* \rangle$ and $\mathbb{G}_2^2 \subseteq \langle \mathbf{v}_1^*, \mathbf{v}_2^* \rangle$, what makes the commitments perfectly hiding. Moreover, for every $\vec{A}, \vec{B}, \vec{X}, \vec{Y}$ there exist $\Delta, \Omega, \Xi, \Upsilon$ such that

$$\iota_1(\vec{A}) = \Delta \mathbf{u}^* \qquad \iota_1(\vec{X}) = \Xi \mathbf{u}^* \qquad \iota_2(\vec{B}) = \Omega \mathbf{v}^* \qquad \iota_2(\vec{Y}) = \Upsilon \mathbf{v}^*$$

We have thus

$$\begin{aligned}\theta &= S^\top \iota_1(\vec{A}) + S^\top \Gamma^\top \iota_1(\vec{X}) + Z\vec{\mathbf{u}}^* &&= \big(S^\top \Delta + S^\top \Gamma^\top \Xi + Z\big)\vec{\mathbf{u}}^* \\ \phi &= R^\top \iota_2(\vec{B}) + R^\top \Gamma \iota_2(\vec{Y}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}^* &&= \big(R^\top \Omega + R^\top \Gamma \Upsilon + R^\top \Gamma S - Z^\top\big)\vec{\mathbf{v}}^*\end{aligned}$$

Since Z is uniformly random we can write $\theta = \Theta \vec{\mathbf{u}}^*$ with Θ uniformly random, and $\phi = \Phi \vec{\mathbf{v}}^*$ where Φ depends on Θ. By perfect completeness, every witness yields a proof with

$$\big(\iota_1(\vec{A}) \bullet \vec{\mathbf{d}}\big) \circ \big(\vec{\mathbf{c}} \bullet \iota_2(\vec{B})\big) \circ \big(\vec{\mathbf{c}} \bullet \Gamma \vec{\mathbf{d}}\big) \circ \big(\iota_T(\mathfrak{t}_T)\big)^{-1} \circ \big(\theta \bullet \vec{\mathbf{v}}^*\big)^{-1} = \big(\vec{\mathbf{u}}^* \bullet \phi\big) \ .$$

Conditioned on Θ every two Φ and Φ' satisfy thus $\mathbf{u}^* \bullet (\Phi - \Phi')\mathbf{v}^* = 0$. It is easily shown that $F(\mathbf{u}_1^*, \mathbf{v}_1^*), F(\mathbf{u}_1^*, \mathbf{v}_2^*), F(\mathbf{u}_2^*, \mathbf{v}_1^*), F(\mathbf{u}_2^*, \mathbf{v}_2^*)$ are linearly independent, which implies that if for some Δ: $\mathbf{u}^* \bullet \Delta \mathbf{v}^* = 0$ then $\Delta = 0$. Thus every Θ determines one single Φ, which means that, since Θ is uniformly chosen, for any witness we get a uniform distribution over (ϕ, θ) conditioned on it being an acceptable proof.

Security. *It follows from the results of* [GS08]*, which we sketched in this section, and Remark 1 that* (Prove, Verify, RdProof) *is a randomizable witness-indistinguishable proof system for* **Com** *from Sect. 2.4.2, as defined in Sect. 2.2.2.*

2.4.5 Batch Verification

While Groth-Sahai proofs are by far the most efficient witness-indistinguishable (and zero-knowledge) proof system that does not resort to random oracles, in particular verification of a proof needs a high amount of computation. While generation of a proof only requires exponentions in the two groups $\mathbb{G}_1$ and $\mathbb{G}_2$, verification requires computation of *pairings*, which is more expensive.

In [7] we show that by using techniques of *batch verification*, the number of pairings required to verify a proof can be reduced considerably.[7] Batch cryptography strives to process expensive tasks in batch rather than individually and was introduced by Fiat [Fia90] and extended in [BGR98]. Ferrara et al. [FGHP09] considered batch verification for pairing-based equations and applied their results to signature verification.

At the cost of introducing a small soundness error it is possible to achieve a high gain in efficiency by verifying k equations at once. If the batch-verification algorithm returns 1 then all equations hold with overwhelming probability. We use the *small exponent test* by Bellare et al. [BGR98]: we pick small random exponents $r_1, \ldots, r_k$, take the i-th equation to the power of r_i and check whether the product of the left-hand sides of the randomized equations equals the product of their right-hand sides. If the pairings in the product equation contain common elements, we can then regroup them to minimize the number of pairings. Moreover, we move exponents "into" the pairing to avoid exponentiations in $\mathbb{G}_T$, which are in practice more expensive then those in $\mathbb{G}_1$ or $\mathbb{G}_2$. Ferrara et al. prove that when the random exponents $r_1, \ldots, r_k$ are ℓ-bit strings then the probability of accepting an *invalid* batch is bounded by $2^{-\ell}$.

Batch Verification of Groth-Sahai Proofs. We apply the batch-verification techniques to the verification equations for Groth-Sahai proofs about pairing-product equations, which are given

[7] Batch verification of Groth-Sahai proofs was independently suggested in [GSW09b]

before Sect. 2.4.4 (p. 23). By grouping pairings in the verification equations, we have already reduced the number of pairings on the left-hand sides of the equation to $3m + 2n$; the right-hand sides contain 16 pairings. To verify all 4 equations in batch, we choose $r_1, r_2, r_3, r_4 \leftarrow \{0,1\}^\ell$ (where ℓ determines the tradeoff between soundness and efficiency), take the i-th equation to the power of r_i and multiply them. Without loss of generality, we assume $n \leq m$ (otherwise we could reorder differently). After rearranging the terms we get the left-hand side

$$\prod_{j=1}^{n} e\left(\left(\prod_{i=1}^{m} c_{i1}^{\gamma_{ij}}\right)^{r_1}\left(A_j \prod_{i=1}^{m} c_{i2}^{\gamma_{ij}}\right)^{r_3},\, d_{j1}\right) \prod_{j=1}^{n} e\left(\left(\prod_{i=1}^{m} c_{i1}^{\gamma_{ij}}\right)^{r_2}\left(A_j \prod_{i=1}^{m} c_{i2}^{\gamma_{ij}}\right)^{r_4},\, d_{j2}\right) \prod_{i=1}^{m} e\left(c_{i1}^{r_2} c_{i2}^{r_4},\, B_i\right)$$

and the right-hand side

$$e(u_{11}, \phi_{11}^{r_1}\phi_{12}^{r_2})\, e(u_{21}, \phi_{21}^{r_1}\phi_{22}^{r_2})\, e(u_{12}, \phi_{11}^{r_3}\phi_{12}^{r_4})\, e(u_{22}, \phi_{21}^{r_3}\phi_{22}^{r_4}) \\ e(\theta_{11}^{r_1}\theta_{12}^{r_3}, v_{11})\, e(\theta_{21}^{r_1}\theta_{22}^{r_3}, v_{21})\, e(\theta_{11}^{r_2}\theta_{12}^{r_4}, v_{12})\, e(\theta_{21}^{r_2}\theta_{22}^{r_4}, v_{22})\, t_T^{r_4}$$

which requires $m + 2n$ pairings and $2mn + 2m + 4n$ exponentiations in $\mathbb{G}_1$ for the left-hand side, and for the right-hand side: 8 pairing computations, 8 exponentiations in $\mathbb{G}_1$, 8 exponentiations in $\mathbb{G}_2$ and one exponentiation in $\mathbb{G}_T$.

If we verify k proofs for the same commitment key $(\vec{\mathbf{u}}, \vec{\mathbf{v}})$ (but in general different commitments) at the same time then we can optimize the computation by verifying them in batch. While the cost of the left-hand side grows linearly in the number of proofs, the right-hand side requires 8 pairings for any k, since we can regroup pairings of the form $e(u_{ij}, \cdot)$ and $e(\cdot, v_{ij})$.

Chapter 3

New Assumptions and their Justifications

Contents

In this chapter we present the new assumptions over bilinear groups on which our constructions of automorphic signatures in Sect. 4 (and thus everything built on top of them) rely. Assumption 7 was introduced in [4], whereas Assumptions 5, 6, and 8 were introduced in [6]. For every assumption we justify its plausibility.

3.1 Flexible CDH

The *Computational Diffie-Hellman* assumption (CDH) in a group $\mathbb{G}$ generated by G states that given G^a and G^b for random integers a and b, it is hard to compute G^{ab}. *Flexible* assumptions give the solver more liberty in that there is not only one solution to an instance. However, they remain generically hard, meaning that there does not exist an efficient algorithm that works for *any* group.

The first new assumption we introduce is a weaker variant of the *1-flexible CDH* assumption [LV08], which is itself a weakening of the *2-out-of-3 CDH* assumption [KJP06]. The latter states that given (G, G^a, G^b), it is hard to output (R, R^{ab}) for an arbitrary non-trivial R (i.e., $R \neq 1$). To solve 1-flexible CDH, one must additionally compute R^a. We weaken the assumption further by defining a solution as (R, R^a, R^b, R^{ab}), and call it the *weak flexible CDH* assumption.

Assumption 5 (WF-CDH)**.** *Given $(G, G^a, G^b) \in \mathbb{G}^3$ for random $a, b \leftarrow \mathbb{Z}_p$, it is hard to output a non-trivial tuple (R, R^a, R^b, R^{ab}), i.e., with $R \in \mathbb{G}^*$.*

In a symmetric bilinear group a solution (R, R^a, R^b, R^{ab}) can be efficiently verified since

$$e(G^a, R^b) = e(R^a, G^b) \qquad e(R^a, G^b) = e(G, R^{ab}) \qquad e(R, G^b) = e(G, R^b)$$

We define a generalization of WF-CDH to *asymmetric* groups. G will be the generator of $\mathbb{G}_1$ and instead of G^b, we give a random generator H of $\mathbb{G}_2$; so a solution $(G^r, G^{ra}, G^{rb}, G^{rab})$ becomes $(G^r, G^{ra}, H^r, H^{rb})$ and can again be efficiently verified due to the pairing.

Assumption 6 (AWF-CDH)**.** *Given random generators $G \in \mathbb{G}_1$ and $H \in \mathbb{G}_2$, and $A = G^a$ for $a \leftarrow \mathbb{Z}_p$, it is hard to output $(G^r, G^{ra}, H^r, H^{ra})$ with $r \neq 0$, i.e., a tuple $(R, M, S, N) \in (\mathbb{G}_1^*)^2 \times (\mathbb{G}_2^*)^2$ that satisfies*

$$e(A, S) = e(M, H) \qquad e(M, H) = e(G, N) \qquad e(R, H) = e(G, S) \tag{3.1}$$

The assumption is easily shown to hold in both generic asymmetric and symmetric bilinear groups. Moreover, if $\mathbb{G}_1 = \mathbb{G}_2$ it becomes WF-CDH, which is implied by the 2-out-of-3 CDH and the 1-flexible CDH assumptions, while in asymmetric groups it is implied by DDH in $\mathbb{G}_1$ (and thus a fortiori by SXDH), as we show now.

Lemma 1. *The AWF-CDH assumption holds in $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ if the decisional Diffie-Hellman problem is hard in $\mathbb{G}_1$.*

Proof. Suppose there exists an efficient algorithm $\mathcal{A}$ that solves the AWF-CDH problem with non-negligible probability. Let (G, G^a, G^b, G^c) be a DDH instance in $\mathbb{G}_1$, i.e., we have to decide whether $c = ab$. We choose $H \leftarrow \mathbb{G}_2$ and run $\mathcal{A}$ on input (G, G^a, H), which, when successful, outputs $(G^r, G^{ar}, H^r, H^{ar})$. We use this to check whether $G^c = G^{ab}$, since $e(G^{ab}, H^r) = e(G^b, H^{ar})$. □

3.2 The DH-SDH Assumption

The *q-strong Diffie-Hellman* (SDH) assumption in a *symmetric* bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ states that given $(G, G^x, G^{(x^2)}, \ldots, G^{(x^q)})$ it is hard to output a pair $(G^{\frac{1}{x+c}}, c)$. Hardness of this problem implies hardness of the following two problems, where G and K are random generators of the group and c_i and v_i are random elements from $\mathbb{Z}_p$.

1. Given G, G^x and $q-1$ pairs $(G^{\frac{1}{x+c_i}}, c_i)$, output a new pair $(G^{\frac{1}{x+c}}, c)$.
2. Given G, K, G^x and $q-1$ triples $\big((K \cdot G^{v_i})^{\frac{1}{x+c_i}}, c_i, v_i\big)$, output a new triple $((K \cdot G^v)^{\frac{1}{x+c}}, c, v)$, i.e., with $(c, v) \neq (c_i, v_i)$ for all i.

The first implication was shown in [BB04], the second one is proved by the following lemma, which appears in [4]:

Lemma 2. *If SDH is hard then so is Problem 2 above.*

Proof. There are two types of solutions (A, c, v) to Problem 2: Either $c \neq c_i$ for all $i = 1, \ldots, q-1$, or $c = c_k$ for some k and $v \neq v_k$. We treat the two types separately:

A New c. Let us first assume that the adversary is more likely to return a solution (A, c, v) with $c \neq c_i$ for all i. Let the $(q+1)$-tuple $(P, P^x, P^{x^2}, \ldots, P^{x^q})$ be an SDH instance. We now generate a random instance of Problem 2. We randomly choose $\alpha, \beta \leftarrow \mathbb{Z}_p$ and $c_i, v_i \leftarrow \mathbb{Z}_p$, for $i = 1, \ldots, q-1$, such that the pairs (c_i, v_i) are pairwise distinct. We then set

$$G := P^{\alpha[\prod_{i=1}^{q-1}(x+c_i)]} \qquad K := G^\beta \qquad X := G^x$$

Since $\prod_{i=1}^{q-1}(x+c_i)$ is a known polynomial in x and since we are given $P^{(x^i)}$ for $i = 0, \ldots, q$ in the SDH instance, we can compute G. The exponent of X in basis P is a known polynomial

of degree q in x; we can thus compute X. Analogously, we can simulate the elements $A_i := (K \cdot G^{v_i})^{\frac{1}{x+c_i}}$, for $i = 1, \ldots, q-1$, for the instance of Problem 2:

$$A_i := P^{\alpha(\beta+v_i)\left[\prod_{\substack{j=1 \\ j\neq i}}^{q-1}(x+c_j)\right]} \ .$$

Since $A_i^{x+c_i} = P^{\alpha(\beta+v_i)\left[\prod_{j=1}^{q-1}(x+c_j)\right]} = G^{\beta+v_i} = K \cdot G^{v_i}$, we have $A_i = (K \cdot G^{v_i})^{\frac{1}{x+c_i}}$.

A successful output (A, c, v) of the first type satisfies $c \neq c_i$ for all i and

$$A = (K \cdot G^{v})^{\frac{1}{x+c}} = G^{\frac{\beta+v}{x+c}} = P^{\frac{\alpha(\beta+v)}{x+c}[\prod_{i=1}^{q-1}(x+c_i)]} = P^{\frac{f(x)}{x+c}} \ ,$$

where f is a polynomial defined as

$$\begin{aligned} f(x) = \alpha(\beta+v)\left[\prod_{i=1}^{q-1}(x+c_i)\right] &= \alpha(\beta+v)\left[\prod_{i=1}^{q-1}((x+c)+(c_i-c))\right] \\ &= \alpha(\beta+v)\left[\prod_{i=1}^{q-1}(c_i-c)\right] + (x+c)g(x+c) \ , \end{aligned}$$

for some (known) polynomial g of degree at most $q-2$. From the SDH instance we can thus compute $B := P^{g(x+c)}$. Since

$$A = P^{\frac{\alpha(\beta+v)\left[\prod_{i=1}^{q-1}(c_i-c)\right]}{x+c}} \cdot P^{g(x+c)} \ ,$$

we can compute

$$A' := (A \cdot B^{-1})^{\frac{1}{\alpha(\beta+v)\left[\prod_{i=1}^{q-1}(c_i-c)\right]}} = P^{\frac{1}{x+c}}$$

which yields a solution (A', c) for the SDH instance.

An Already Known c. Let us now assume that the adversary most likely returns a solution (A, c, v) with $c \in \{c_1, \ldots, c_{q-1}\}$ from the input. Again, let us be given an SDH instance $(P, P^x, P^{x^2}, \ldots, P^{x^q})$. We generate a random instance of Problem 2: we randomly choose $\alpha, \beta \leftarrow \mathbb{Z}_p$ and $(c_i, v_i) \leftarrow \mathbb{Z}_p$, for $i = 1, \ldots, q-1$, such that the pairs (c_i, v_i) are pairwise distinct; moreover, we choose a random index $k \leftarrow \{1, \ldots, q-1\}$, and set

$$G := P^{\beta\left[\prod_{\substack{i=1 \\ i\neq k}}^{q-1}(x-c_k+c_i)\right]} \qquad K := G^{\alpha x - v_k} \qquad X := G^x \cdot G^{-c_k} = G^y$$

which implicitly defines a new secret exponent $y := x - c_k$. We now have to simulate the A_i for the instance; we distinguish two cases.

- For $i = k$, we set $A_k := G^{\alpha}$, since then $A_k^{y+c_k} = G^{\alpha x} = K \cdot G^{v_k}$.
- For $i \neq k$, we set

$$A_i := P^{\beta((v_i-v_k)+\alpha x)\left[\prod_{\substack{j=1 \\ j\neq i,k}}^{q-1}(x-c_k+c_j)\right]} \ ,$$

  and have $A_i^{y+c_i} = G^{(v_i-v_k)+\alpha x} = G^{v_i-v_k} \cdot K \cdot G^{v_k} = K \cdot G^{v_i}$.

Thus, in both cases we simulated A_i correctly.

A successful output satisfies $(c, v) \neq (c_i, v_i)$ for all $i = 1, \ldots, q$, and $A = (K \cdot G^{v})^{\frac{1}{y+c}}$. Since we assumed that $c \in \{c_1, \ldots, c_q\}$, and c_k is only used in the form of $y = x - c_k$, where x was uniformly random, the probability that $c = c_k$ is $\frac{1}{q}$. In this case we have $v \neq v_k$ and

$$A = (K \cdot G^{v})^{\frac{1}{x}} \qquad A_k = (K \cdot G^{v_k})^{\frac{1}{x}}$$

and thus

$$A^{v_k} \cdot A_k^{-v} = K^{\frac{v_k - v}{x}} = P^{\frac{v_k - v}{x}(\alpha x - v_k)\beta\left[\prod_{\substack{i=1 \\ i \neq k}}^{q-1}(x - c_k + c_i)\right]} = P^{\frac{f(x)}{x}}$$

with

$$f(x) = (v_k - v)(\alpha x - v_k)\beta\Big[\prod_{\substack{i=1 \\ i \neq k}}^{q-1}(x - c_k + c_i)\Big] = (v - v_k)v_k\,\beta\Big[\prod_{\substack{i=1 \\ i \neq k}}^{q-1}(c_i - c_k)\Big] + xg(x) \ ,$$

for a (known) polynomial g of degree at most $q-2$. We can thus compute $B := P^{g(x)}$ from the SDH instance. Since

$$A^{v_k} \cdot A_k^{-v} = P^{\frac{1}{x}(v - v_k)v_k\,\beta\left[\prod_{\substack{i=1 \\ i \neq k}}^{q-1}(c_i - c_k)\right] + g(x)} \ .$$

setting

$$A' = (A^{v_k} \cdot A_k^{-v} \cdot B^{-1})^{\frac{1}{(v - v_k)v_k\beta\left[\prod_{\substack{i=1 \\ i \neq k}}^{q-1}(c_i - c_k)\right]}} = P^{\frac{1}{x}}$$

yields a solution $(A', 0)$ for the given SDH instance. □

Boyen and Waters [BW07] define the *hidden* SDH (HSDH) assumption which states that Problem 1 is hard when the pairs $(G^{1/(x+c_i)}, c_i)$ in the instance and the solution are substituted with triples of the form $(G^{1/(x+c_i)}, G^{c_i}, H^{c_i})$, for a fixed H (see Sect. 2.3, p. 18). A triple (A, C, D) of this form is efficiently decidable since it satisfies the following equations:

$$e(A, X \cdot C) = e(G, G) \qquad\qquad e(C, H) = e(G, D)$$

HSDH is incomparable to SDH: an HSDH instance can be computed from an SDH instance and an HSDH solution can be computed from an SDH solution, but not the other way round.[1]

Analogously, we define the *double hidden* SDH (DH-SDH) assumption by "hiding" the scalars of Problem 2: instead of directly giving (and requiring in a solution) c_i and v_i, we substitute them with exponentiations of two group elements:

Assumption 7 (q-DH-SDH)**.** *In a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, G)$, given $(H, K, X = G^x) \in \mathbb{G}^3$ and $q-1$ tuples*

$$\big(A_i = (K \cdot G^{v_i})^{\frac{1}{x+c_i}},\ C_i = G^{c_i},\ D_i = H^{c_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\big)$$

for random $c_i, v_i \leftarrow \mathbb{Z}_p$, it is hard to output a new tuple $(A, C, D, V, W) \in \mathbb{G}^5$ of this form.

Note that a tuple (A, C, D, V, W) of this form satisfies the following equations:

$$e(A, X \cdot C) = e(K \cdot V, G) \qquad e(C, H) = e(G, D) \qquad e(V, H) = e(G, W) \tag{3.2}$$

An additional justification of the DH-SDH assumption is the fact that under the *Knowledge-of-Exponent assumption* (KEA) [Dam92, BP04], hardness of q-DHSDH follows directly from hardness of Problem 2, which is implied by SDH. KEA asserts that given (G, H), from an adversary returning (G^{c^*}, H^{c^*}) and (G^{v^*}, H^{v^*}) one can extract c^* and v^*.

[1] Note that BB-HSDH, given in Assumption 9 on p. 31 is trivially stronger than both SDH and HSDH.

3.3 The ADH-SDH Assumption

A quintuple of DH-SDH can only be efficiently verified in symmetric bilinear groups. We now generalize DH-SDH to asymmetric bilinear groups (*A*DH-SDH), which will allow for more flexibility in the choice of groups and will lead to a more efficient instantiation of automorphic signatures in Chapter 4. The element H will now be chosen from $\mathbb{G}_2$ and the other generators from $\mathbb{G}_1$; we add one more generator $F \in \mathbb{G}_1$ and instead of the elements $C_i = G^{c_i}$ we give $B_i = F^{c_i}$. This makes it possible to include $Y = H^x$ which allows efficient verification of a tuple due to the pairing (if we also gave G^{c_i}, we arrive at an easy problem; see Sect. 3.3.1).

Assumption 8 (q-ADH-SDH)**.** *In an asymmetric group let $G, F, K \in \mathbb{G}_1$, $H \in \mathbb{G}_2$ and $x, c_i, v_i \in \mathbb{Z}_p$ be random. Given $(G, F, K, X = G^x;\ H, Y = H^x)$ and*

$$\left(A_i = (K \cdot G^{v_i})^{\frac{1}{x+c_i}},\ B_i = F^{c_i},\ D_i = H^{c_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right)\ ,$$

for $1 \le i \le q-1$, it is hard to output a tuple $((K \cdot G^v)^{\frac{1}{x+c}}, F^c, H^c, G^v, H^v)$ with $(c, v) \neq (c_i, v_i)$ for all i.

Note that a tuple (A, B, D, V, W) of this form satisfies the following equations:

$$e(A, Y \cdot D) = e(K \cdot V, H) \qquad e(B, H) = e(F, D) \qquad e(V, H) = e(G, W) \tag{3.3}$$

Assumption 8 is also valid in generic *symmetric* bilinear groups; in particular, in Sect. 3.3.2 we prove generic security of ADH-SDH in the symmetric setting (thus, a fortiori it holds when $\mathbb{G}_1 \neq \mathbb{G}_2$).

3.3.1 A Note on ADH-SDH

One could be tempted to transfer the DH-SDH assumption to asymmetric groups by adding $Y := H^{\log_G X}$ to the instance, which would allow to check validity of a tuple (A, C, V, D, W). However, this assumption is wrong, as it succumbs to the following attack: Given an instance $(G, H, K, X, Y, (A_i, C_i, V_i, D_i, W_i)_{i=1}^{q-1})$, set $A^* := A_1^{-1}$, $C^* := X^{-2} \cdot C_1^{-1}$, $D^* := Y^{-2} \cdot D_1^{-1}$, $V^* := V_1$, $W^* := W_1$. Then we have $e(A^*, Y \cdot D^*) = e(A_1^{-1}, (Y \cdot D_1)^{-1}) = e(K \cdot V_1, H) = e(K \cdot V^*, H)$. The attack comes from the fact that we can use X and Y to simultaneously build C^* and D^*. This is what makes it indispensable to use a different basis for the C, leading to a generically secure assumption, as proved in the next section.

The q-ADH-SDH assumption is quite similar to the q-BB-HSDH assumption introduced in [BCC+09], which is another hidden variant of SDH, where both G^x and H^x are given.

Assumption 9 (BB-HSDH)**.** *Let $x, c_1, \ldots, c_{q-1} \leftarrow \mathbb{Z}_p$. Then on input $G, G^x, F \in \mathbb{G}_1$ and $H, H^x \in \mathbb{G}_2$ and tuples $(G^{\frac{1}{x+c_i}}, c_i)_{i=1}^{q-1}$, it is infeasible to output a tuple $(G^{\frac{1}{x+c}}, F^c, H^c)$ with $c \neq c_i$ for all i.*

BB-HSDH is however incomparable to ADH-SDH, since while ADH-SDH gives the adversary more flexibility in his output, BB-HSDH gives him more information as input, since the c_i are given explicitly. Moreover, BB-HSDH is somehow asymmetric, in that the task is to output a tuple that is easier to construct than a tuple that has the form of the $q-1$ input tuples. This is why it is stronger than both HSDH *and* hardness of Problem 1 on p. 28. Note that if we had $F = G$ (as in the original definition of HSDH in [BW07]), the BB-HSDH problem would become easy as the attack sketched above applies here as well.

3.3.2 Generic Security of the q-ADH-SDH Assumption

In this section we will prove that ADH-SDH is hard in the *generic group model* [Sho97] for symmetric bilinear groups (which implies the asymmetric case). This means that no adversary can break the assumption using only operations for a generic bilinear group. We model a generic group by representing each group element as a random encoding and providing the adversary with access to oracles modelling the group operations $\mathbb{G}$ and $\mathbb{G}_T$ as well as the pairing $e\colon \mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$. We define $[\cdot]\colon \mathbb{Z}_p\to\mathbb{G}$ and $[[\cdot]]\colon \mathbb{Z}_p\to\mathbb{G}_T$ and give the adversary access to the group elements as encodings of their discrete logarithms: when queried $(\texttt{exp}, x)$, the oracle returns $[x]$; on $(\texttt{mult}, [x], [y])$ and $(\texttt{mult}, [[x]], [[y]])$ it returns $[x+y]$ and $[[x+y]]$, respectively; finally, a query $(\texttt{pair}, [x], [y])$ is answered with $[[xy]]$. When given a problem instance consisting of group elements $[x_1], \ldots, [x_n]$, all the adversary can do with the help of the oracle is generate encodings of low-degree polynomials in $\mathbb{Z}_p[x_1, \ldots, x_n]$.

To solve the problem the adversary must output the encoding of a solution. The Schwartz-Zippel theorem [Sch80] states that the probability of a non-zero low-degree polynomial to evaluate to 0 for randomly chosen $x_1, \ldots, x_n$ is negligible. We thus have to show that the solution is not expressible as a linear combination of the input polynomials. For convenience we restate the assumption for symmetric bilinear groups.

(q-ADH-SDH) Given $(G, F, H, K, X=G^x, Y=H^x) \in \mathbb{G}^6$ and $q-1$ tuples

$$\left(A_i = (K\cdot G^{v_i})^{\frac{1}{x+c_i}},\ B_i = F^{c_i},\ D_i = H^{c_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right),$$

with $c_i, v_i \leftarrow \mathbb{Z}_p^*$ for $i = 1, \ldots, q-1$, it is hard to output a new tuple $(A^*, B^*, D^*, V^*, W^*)$ that satisfies

$$e(A^*, Y\cdot D^*) = e(K\cdot V^*, H) \qquad e(B^*, H) = e(F, D^*) \qquad e(V^*, H) = e(G, W^*) \tag{3.4}$$

Theorem 1. *The q-ADH-SDH assumption holds in generic bilinear groups when q is a polynomial.*

Proof. We work with the "discrete-log" representation of all group elements w.r.t. basis G. A q-ADH-SDH instance is thus represented by the following rational fractions (each lower-case letter denotes the logarithm of the group elements denoted by the corresponding upper-case letter):

$$1,\ f,\ h,\ k,\ x,\ y = xh,\ \left\{a_i = \tfrac{k+v_i}{x+c_i},\ b_i = c_i f,\ d_i = c_i h,\ v_i,\ w_i = v_i h\right\}_{i=1}^{q-1} \tag{3.5}$$

Considering the logarithms of the $\mathbb{G}_T$-elements in (3.4) w.r.t. the basis $e(G, G)$ yields

$$a^*(xh + d^*) = (k + v^*)h \qquad b^*h = d^*f \qquad v^*h = w^* \tag{3.6}$$

In a generic group, all the adversary can do is apply the group operation to the elements of its input. We will show that the only linear combinations $(a^*, b^*, d^*, v^*, w^*)$ of elements in (3.5) satisfying (3.6) are $(a^* = a_i = \frac{k+v_i}{x+c_i}, b^* = b_i = c_i f, d^* = d_i = c_i h, v^* = v_i, w^* = w_i = v_i h)$ for some i; which means all the adversary can do is return a quintuple from the instance. We make the following ansatz for a^* (and analogously for b^*, d^*, v^* and w^*):

$$a^* = \alpha + \alpha_f f + \alpha_h\, h + \alpha_k\, k + \alpha_x\, x + \alpha_y\, xh + \sum \alpha_{a_i} \tfrac{k+v_i}{x+c_i} + \sum \alpha_{b,i}\, c_i f + \sum \alpha_{d,i}\, c_i h + \sum \alpha_{v,i}\, v_i + \sum \alpha_{w,i}\, v_i h$$

Since for any v^* the adversary forms, it has to provide v^*h as well, we can limit the elements used for v^* to those of which their product with h is also given: $1, x$ and v_i (for all i). Similarly, plugging

in the ansätze for b^* and d^* in the second equation of (3.6) and equating coefficients eliminates most of the coefficients. Thus, the last two equations of (3.6) simplify b^*, d^*, v^* and w^* to

$$b^* = \gamma_f f + \sum \gamma_{b,i}\, c_i f \qquad v^* = \mu + \mu_x x + \sum \mu_{v,i} v_i$$
$$d^* = \gamma_f\, h + \sum \gamma_{b,i}\, c_i h \qquad w^* = \mu h + \mu_x x h + \sum \mu_{v,i} v_i h$$

We substitute a^*, d^*, v^* by their ansätze in the first equation of (3.6), that is $a^*(xh+d^*)-v^*h = kh$. After some rearranging we get (for convenience, we omit one h per term, i.e., we symbolically "divided" the equation by h):

$$(\alpha\gamma_f - \mu)\ 1 \ +\ (\alpha_f\gamma_f)\ f \ +\ (\alpha_h\gamma_f)\ h \ +\ (\alpha + \alpha_x\gamma_f - \mu_x)\ x \ +\ (\alpha_h + \alpha_y\gamma_f)\ xh \ + \tag{3.7a}$$

$$\sum (\alpha_{a,i}\gamma_f)\ \tfrac{k+v_i}{x+c_i} \ +\ \sum (\alpha_{b,i}\gamma_f + \alpha_f\gamma_{b,i})\ c_i f \ +\ \sum (\alpha_{d,i}\gamma_f + \alpha_h\gamma_{b,i})\ c_i h \ +\ \sum (\alpha_{w,i}\gamma_f)\ v_i h \ + \tag{3.7b}$$

$$(\alpha_f)\ xf \ +\ (\alpha_k)\ xk \ +\ (\alpha_x)\ x^2 \ +\ (\alpha_y)\ x^2 h \ +\ \sum (\alpha_{d,i} + \alpha_y\gamma_{b,i})\ c_i xh \ +\ \sum (\alpha_{b,i})\, c_i xf \ + \tag{3.7c}$$

$$\sum (\alpha_{v,i})\ v_i x \ +\ \sum (\alpha_{w,i})\ v_i xh \ +\sum (\alpha\gamma_{b,i})\, c_i \ +\ \sum (\alpha_k\gamma_{b,i})\, c_i k \ +\ \sum (\alpha_x\gamma_{b,i})\, xc_i \ + \tag{3.7d}$$

$$\sum\sum (\alpha_{b,i}\gamma_{b,j})\, c_i c_j f \ +\ \sum\sum (\alpha_{d,i}\gamma_{b,j})\, c_i c_j h \ +\ \sum\sum (\alpha_{v,i}\gamma_{b,j})\, v_i c_j \ +\ \sum\sum (\alpha_{w,i}\gamma_{b,j})\, v_i c_j h \ + \tag{3.7e}$$

$$\underbrace{(\alpha_k\gamma_f)}_{=:\lambda_k}\ k \ +\ \sum \underbrace{(\alpha_{v,i}\gamma_f - \mu_{v,i})}_{=:\lambda_{v,i}}\ v_i \ +\ \sum \underbrace{(\alpha_{a,i})}_{=:\lambda_{xa,i}}\ \tfrac{x(k+v_i)}{x+c_i} \ +\ \sum\sum \underbrace{(\alpha_{a,i}\gamma_{b,j})}_{=:\lambda_{ca,i,j}}\ \tfrac{c_j(k+v_i)}{x+c_i} \quad = \quad k \tag{3.7f}$$

Comparison of coefficients[2] of the two sides of the equation shows that all coefficients in lines (3.7a)–(3.7e) must be 0, whereas for the last line we have a different situation: adding $\frac{x(k+v_i)}{x+c_i}$ and $\frac{c_i(k+v_i)}{x+c_i}$ reduces to $k+v_i$ (but this is the only combination that reduces); we have thus

$$\text{for all } i: \lambda_{xa,i} = \lambda_{ca,i,i} \qquad \text{for all } i \neq j: \lambda_{ca,i,j} = 0 \tag{3.8}$$
$$\text{coefficient of } k\text{:}\quad \sum \lambda_{xa,i} + \lambda_k = 1 \qquad \text{coefficient of } v_i\text{:}\quad \lambda_{xa,i} + \lambda_{v,i} = 0 \tag{3.9}$$

We now solve the equations "all coefficients in Lines (3.7a) to (3.7e) equal 0", and Equations (3.8) and (3.9) for the values $\big(\alpha, \alpha_f, \alpha_h, \alpha_k, \alpha_x, \alpha_y, \gamma_f, \mu, \mu_x, \{\alpha_{a,i}, \alpha_{b,i}, \alpha_{d,i}, \alpha_{v,i}, \alpha_{w,i}, \gamma_{b,i}, \mu_{v,i}\}\big)$:

The first four terms and the last term in Line (3.7c) and the first two terms in Line (3.7d) immediately yield: $\alpha_f = \alpha_k = \alpha_x = \alpha_y = \alpha_{b,i} = \alpha_{v,i} = \alpha_{w,i} = 0$ for all i. Now $\alpha_y = 0$ implies $\alpha_h = 0$ by the last term in (3.7a), and $\alpha_y = 0$ implies $\alpha_{d,i} = 0$ for all i by the fifth term in (3.7c). Plugging in these values, the only equations different from "$0 = 0$" are the following:

$$\alpha\,\gamma_f - \mu = 0 \qquad \alpha - \mu_x = 0 \tag{3.10}$$
$$\alpha_{a,i}\,\gamma_f = 0 \ \ (\forall i) \qquad \alpha\,\gamma_{b,i} = 0 \ \ (\forall i) \tag{3.11}$$
$$\alpha_{a,i}(1 - \gamma_{b,i}) = 0 \ \ (\forall i) \qquad \alpha_{a,i}\,\gamma_{b,j} = 0 \ \ (\forall i \neq j) \tag{3.12}$$
$$\textstyle\sum_{i=1}^{q-1} \alpha_{a,i} = 1 \qquad \alpha_{a,i} - \mu_{v,i} = 0 \ \ (\forall i) \tag{3.13}$$

[2] To do straightforward comparison of coefficients, we actually would have to multiply the equation by $\prod_{i=1}^{q-1}(x+c_i)$ first. For the sake of presentation, we keep the fractions and instead introduce new equations for the cases where a linear combination leads to a fraction that cancels down.

where the second equation in (3.10) "(3.10.2)" follows from the fourth term in (3.7a) and $\alpha_x = 0$. (3.11.1) and (3.11.2) follow from the first term in (3.7b) and the third term in (3.7d), respectively. Equations (3.12) are the equations in (3.8); and those in (3.13) are the ones from (3.9) taking into account that $\alpha_k = 0$ and $\alpha_{v,i} = 0$ for all i. The variables not yet proved to be 0 are $\alpha, \gamma_f, \mu, \mu_x, \alpha_{a,i}, \gamma_{b,i}$ and $\mu_{v,i}$ for $1 \leq i \leq q-1$.

We first show that there exists $i^* \in \{1, \ldots, q-1\}$ such that $\alpha_{a,j} = 0$ for all $j \neq i^*$: assume there exist $i \neq j$ such that $\alpha_{a,i} \neq 0$ and $\alpha_{a,j} \neq 0$; then by (3.12.1) we have $\gamma_{b,i} = \gamma_{b,j} = 1$, which contradicts (3.12.2).

This result implies the following: by (3.13.1) we have $\alpha_{a,i^*} = 1$ and by (3.12.1) we have $\gamma_{b,i^*} = 1$, whereas for all $j \neq i^*$: $\gamma_{b,j} = 0$ by (3.12.2). We have thus shown that $\alpha_{a,i^*} = \gamma_{b,i^*} = 1$ and $\alpha_{a,j} = \gamma_{b,j} = 0$ for all $j \neq i^*$.

This now implies $\alpha = 0$ (by (3.11.2)) and thus $\mu = \mu_x = 0$ (by (3.10.1) and (3.10.2), respectively). Moreover $\gamma_f = 0$ (by (3.11.1)) and for all i: $\alpha_{a,i} = \mu_{v,i}$ (by (3.13.2)). The only non-zero variables are thus $\alpha_{a,i^*} = \gamma_{b,i^*} = \mu_{v,i^*} = 1$.

Plugging in our results in the ansätze for a^*, b^*, d^*, v^* and w^*, we proved that there exists $i^* \in \{1, \ldots, q-1\}$ such that $a^* = \frac{k+v_{i^*}}{x+c_{i^*}}$, $b^* = c_{i^*} f$, $d^* = c_{i^*} h$, $v^* = v_{i^*}$ and $w^* = v_{i^*} h$. This means that the only tuples $(A^*, B^*, D^*, V^*, W^*)$ satisfying (3.4) and being generically constructable from a ADH-SDH instance are the tuples from that instance, which concludes our proof of generic security of ADH-SDH. □

Chapter 4

Automorphic Signatures

Contents

In this chapter we instantiate the concept of automorphic signatures in bilinear groups discussed in Sect. 1.1 (p. 4). We then use one of the schemes to construct a blind signature scheme which was the first efficient scheme with a round-optimal issuing protocol; the scheme is thus concurrently secure. Finally, we give generic transformations of a signature scheme to a scheme signing several messages at once. The results of this chapter appear in [6]. We start with a formal definition of automorphic signatures.

Definition 1. *An* automorphic signature *over a bilinear group* $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G_1, G_2)$ *is an EUF-CMA secure signature whose verification keys are contained in the message space. Moreover, the messages and signatures consist of elements of* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*, and the verification predicate is a conjunction of pairing-product equations over the verification key, the message and the signature.*

4.1 Instantiations

To give a first instantiation of automorphic signatures we start with the following observation: DH-SDH (Assumption 7, p. 30) states that given $G, K, H, X = G^x$ and tuples of the form

$$\left(A_i = (K \cdot G^{v_i})^{\frac{1}{x+c_i}},\ C_i = G^{c_i},\ D_i = H^{c_i},\ V_i = G^{v_i},\ W_i = H^{v_i}\right) ,$$

it is hard to compute a new such tuple, which immediately yields a weakly secure signature scheme: consider G, K and H as parameters, X as the verification key, x as the signing key, (V_i, W_i) as a messages in $\mathcal{DH} = \{(G^v, H^v) \,|\, v \in \mathbb{Z}_p\}$, and (A_i, C_i, D_i) as signatures. Given a message (V, W), a signer holding the secret key x can choose c and produce the corresponding (A, C, D) without knowing v.[1] DH-SDH then means that given signatures on random messages, it is hard to find a signature on a new message.

We show how to transform this into a CMA-secure signature scheme by assuming WF-CDH. We add some more randomness to the signature that lets us map a query for a message chosen

[1] Note that this is not the case for the q-HSDH assumption (Assumption 4, p. 18): we cannot regard (G^c, H^c) as the message, since the signer must know c in order to produce $G^{\frac{1}{x+c}}$. If the message is a public key, whose exponent is the secret key, the latter usually cannot be given to the signer, which is precisely the reason for the complex protocol in [BCC+09].

by the adversary to a given tuple $(A_i, C_i, D_i, V_i, W_i)$ from a DH-SDH instance. WF-CDH then asserts that the adversary cannot produce a signed new message $\big((A^*, C^*, D^*, R^*, S^*), (M^*, N^*)\big)$ that maps back to a tuple from the instance (see the proof of Theorem 3 below).

Scheme 1 ($\mathbf{Sig}_1$). Setup_1 *is given a symmetric bilinear group* $(p, \mathbb{G}, \mathbb{G}_T, e, G)$ *and chooses parameters* $(H, K, T) \leftarrow \mathbb{G}^3$*, which define the message space as* $\mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$.

KeyGen_1 *chooses a secret key* $x \leftarrow \mathbb{Z}_p$ *and sets* $\mathsf{vk} := G^x$.

$\mathsf{Sign}_1(x, (M, N))$ *signs a message* $(M, N) \in \mathcal{DH}$ *by choosing* $c, r \leftarrow \mathbb{Z}_p$ *and outputting*

$$\big(A := (K \cdot T^r \cdot M)^{\frac{1}{x+c}},\, C := G^c,\, D := H^c,\, R := G^r,\, S := H^r\big)\ .$$

Ver_1 *accepts a signature* (A, C, D, R, S) *on a message* $(M, N) \in \mathcal{DH}$ *for public key* X *if it satisfies*

$$e(A, X \cdot C) = e(K \cdot M, G)\, e(T, R) \qquad e(C, H) = e(G, D) \qquad e(R, H) = e(G, S) \tag{4.1}$$

Theorem 2. *Under q-DH-SDH and WF-CDH,* $\mathbf{Sig}_1$ *is strongly existentially unforgeable against adversaries making at most* $q - 1$ *adaptive chosen-message queries.*

The proof is analogous to that of Theorem 3 below.

Remark 2. (1) The above scheme can be easily extended to a *certified signature scheme* [BFPW07]. In such a scheme users let their public keys be certified by an authority. A certified signature consists of the user public key, the certificate on it and a signature on the message under the user public key. Given certified signatures on chosen messages for various public keys, it must be hard to produce a new certified signature (either with a new or a given user key).

Consider two instances of $\mathbf{Sig}_1$ (one for certification, one for signatures) that share parameters G, K and T but use a different H_i each. The certification authority's key is G^x, user public keys are of the form (G^v, H_1^v) and a certified signature on a message (G^m, H_2^m) is of the form

$$\big(((K \cdot T^r \cdot G^v)^{\frac{1}{x+c}}, G^c, H_1^c, G^r, H_1^r),\ (G^v, H_1^v),\ (K \cdot T^s \cdot G^m)^{\frac{1}{v+d}}, G^d, H_2^d, G^s, H_2^s)\big)\ .$$

Security follows analogously to the next construction:

(2) From the certified signature we can construct an automorphic scheme $\mathbf{Sig}_{1+1}$ as follows.[2] The public key is a certification-authority key extended to (G^x, H_2^x). An automorphic signature on a message (G^m, H_2^m) is produced by generating a random user key (G^v, H_1^v), certifying it and making a certified signature on the message under that key.

Public keys of $\mathbf{Sig}_{1+1}$ are thus contained in the message space. Security follows from the following hybrid argument. Forgeries using a new one-time key (G^v, H_1^v) are reduced to forgeries for the 1^{st}-level scheme (the simulator gets a challenge $\mathbf{Sig}_1$ key X, chooses $h \leftarrow \mathbb{Z}_p$, sets $H_2 := G^h$ and can thus produce a $\mathbf{Sig}_{1+1}$ key (X, X^h) from X). Forgeries recycling a key from a signing query are reduced security of the 2^{nd}-level scheme (the simulator sets $H_1 := G^h$, guesses the recycled key (G^v, H_1^v) and sets it to (X, X^h) with X a challenge public key of the 2^{nd}-level scheme). A signature consists of 12 group elements satisfying 7 PPEs (of which 5 are linear).

In the asymmetric setting (or assuming ADH-SDH rather than DH-SDH in symmetric groups), we get the following more efficient construction, whose signatures are in $\mathbb{G}_1^3 \times \mathbb{G}_2^2$. Note that the scheme can also be instantiated for $\mathbb{G}_1 = \mathbb{G}_2$. The scheme is similar to Scheme 1 but since an ADH-SDH instance contains G^x and H^x, the public keys lie in the message space, which avoids the hybrid construction from Remark 2 to make it automorphic.

[2] More generally, this way one could transform any certified-signature scheme whose authority keys lie in the message space into an automorphic signature scheme.

Scheme 2 (Sig). Setup. *Given* $grp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$*, choose additional generators* $F, K, T \leftarrow \mathbb{G}_1$*. The message space containing the public key space is* $\mathcal{DH} := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p\}$*.*

KeyGen. *Choose* $sk = x \leftarrow \mathbb{Z}_p$ *and set* $vk = (G^x, H^x)$

Sign. *A signature on* $(M, N) \in \mathcal{DH}$*, valid under public key* (G^x, H^x)*, is defined as*

$$\left(A := (K \cdot T^r \cdot M)^{\frac{1}{x+c}},\, B := F^c,\, D := H^c,\, R := G^r,\, S := H^r\right) , \quad \text{for random } c, r \leftarrow \mathbb{Z}_p$$

Ver. (A, B, D, R, S) *is valid on a message* $(M, N) \in \mathcal{DH}$ *under a public key* $vk = (X, Y) \in \mathcal{DH}$ *iff*

$$e(A, Y \cdot D) = e(K \cdot M, H)\, e(T, S) \qquad e(B, H) = e(F, D) \qquad e(R, H) = e(G, S) \tag{4.2}$$

Theorem 3. *Assuming q-ADH-SDH and AWF-CDH,* **Sig** *is strongly existentially unforgeable against adversaries making at most* $q - 1$ *adaptive chosen-message queries.*

Proof. Consider an adversary that after receiving parameters (G, F, K, T, H) and public key (X, Y) is allowed to ask for $q - 1$ signatures $(A_i, B_i, D_i, R_i, S_i)$ on messages $(M_i, N_i) \in \mathcal{DH}$ of its choice and outputs $(M, N) \in \mathcal{DH}$ and a valid signature (A, B, D, R, S) on it, such that either (M, N) was never queried, or $(M, N) = (M_i, N_i)$ and $(A, B, D, R, S) \neq (A_i, B_i, D_i, R_i, S_i)$. We distinguish two kinds of forgers: An adversary is called of Type I if its output satisfies the following

$$\forall\, 1 \leq i \leq q - 1 : \left[e(T, S \cdot S_i^{-1}) \neq e(M_i \cdot M^{-1}, H) \;\vee\; B \neq B_i\right] \;; \tag{4.3}$$

otherwise it is called of Type II. We will use the first type to break q-ADH-SDH and the second type to break AWF-CDH. For convenience we restate the verification relations for a ADH-SDH solution from (3.3)

$$e(A, Y \cdot D) = e(K \cdot V, H) \qquad e(B, H) = e(F, D) \qquad e(V, H) = e(G, W)$$

and those for a AWF-CDH solution from (3.1)

$$e(A, S) = e(M, H) \qquad e(M, H) = e(G, N) \qquad e(R, H) = e(G, S)$$

Type I Let $\left(G, F, K, X, H, Y, (A_i, B_i, V_i, D_i, W_i)_{i=1}^{q-1}\right)$ be a q-ADH-SDH challenge. It thus satisfies

$$e(A_i, Y \cdot D_i) = e(K \cdot V_i, H) \qquad e(B_i, H) = e(F, D_i) \qquad e(V_i, H) = e(G, W_i) \tag{4.4}$$

Let $\mathcal{A}$ be a forger of Type I. Choose $t \leftarrow \mathbb{Z}_p$ and give parameters $(G, F, K, T := G^t, H)$ and the public key (X, Y) to $\mathcal{A}$. The i-th query for $(M_i, N_i) \in \mathcal{DH}$ is answered as

$$\left(A_i, B_i, D_i, R_i := (V_i \cdot M_i^{-1})^{\frac{1}{t}}, S_i = (W_i \cdot N_i^{-1})^{\frac{1}{t}}\right) .$$

It is easily verified that it satisfies (4.2); for the first equation we have $W_i = S_i^t \cdot N_i$ and thus $e(G, W_i) = e(T, S_i)\, e(G, N_i) = e(T, S_i)\, e(M_i, H)$ since $(M, N) \in \mathcal{DH}$. This implies that $e(K \cdot M_i, H)\, e(T, S_i) = e(K, H)\, e(G, W_i) \overset{(4.4)}{=} e(K \cdot V_i, H) \overset{(4.4)}{=} e(A_i, Y \cdot D_i)$. Moreover, $(A_i, B_i, D_i, R_i, S_i)$ is correctly distributed since v_i is uniformly random in the ADH-SDH instance.

If the adverseray produces a valid signature/message pair $((A, B, D, R, S), (M, N))$ then by the last 2 equations of (4.2), there exist c, r s.t. $B = F^c, D = H^c, R = G^r, S = H^r$, and

$$e(A, Y \cdot D) = e(K \cdot M, H)\, e(T, S) \;. \tag{4.5}$$

The tuple $(A, B, D, V := R^t \cdot M, W := S^t \cdot N)$ satisfies (3.3), since (B, D) and (V, W) are Diffie-Hellman pairs and $e(K \cdot V, H) = e(K \cdot (G^r)^t \cdot M, H) = e(K \cdot M, H)\, e(T, S) \overset{(4.5)}{=} e(A, Y \cdot D)$. Moreover, it is a solution for the ADH-SDH instance, since it is a *new* tuple: assume that for some i we have $B = B_i$ and $W = W_i$, that is $S^t \cdot N = S_i^t \cdot N_i$. Since $(M, N), (M_i, N_i) \in \mathcal{DH}$, we have $e(T, S)\, e(M, H) = e(T, S)\, e(G, N) = e(G, S^t \cdot N) = e(G, S_i^t \cdot N_i) = e(T, S_i)\, e(G, N_i) = e(T, S_i)\, e(M_i, H)$. We have thus $e(T, S \cdot S_i^{-1}) = e(M_i \cdot M^{-1}, H)$ and $B = B_i$ which contradicts (4.3) and thus the fact that $\mathcal{A}$ is of Type I.

Type II Let $(G, H, T = G^t)$ be an AWF-CDH instance; let $\mathcal{A}$ be a forger of Type II. Pick $F, K \leftarrow \mathbb{G}_1$ and $x \leftarrow \mathbb{Z}_p$, set $X := G^x$, $Y := H^x$ and give the adversary parameters (G, F, K, T, H) and public key (X, Y). Answer a signing query on $(M_i, N_i) \in \mathcal{DH}$ by returning a signature $(A_i, B_i, D_i, R_i, S_i)$ produced by $\mathsf{Sign}(x, \cdot)$. Suppose $\mathcal{A}$ returns $((A, B, D, R, S), (M, N))$ satisfying (4.2) s.t. for some i

$$e(T, S \cdot S_i^{-1}) = e(M_i \cdot M^{-1}, H) \qquad\qquad B = B_i \tag{4.6}$$

Then $(M^* := M_i \cdot M^{-1}, N^* := N_i \cdot N^{-1}, R^* := R \cdot R_i^{-1}, S^* := S \cdot S_i^{-1})$ is a AWF-CDH solution: it satisfies the respective equations in (3.1), since (M^*, N^*) and (R^*, S^*) are both DH pairs and $e(T, S^*) = e(T, S \cdot S_i^{-1}) \overset{(4.6)}{=} e(M_i \cdot M^{-1}, H) = e(M^*, H)$. Moreover, (M^*, N^*, R^*, S^*) is non-trivial: if $M^* = 1 = R^*$ then $M = M_i$ and $R = R_i$; by (4.6) we also have $B = B_i$ and since the values M, B and R completely determine A, D, S and N, and thus a message/signature pair, this means that $\mathcal{A}$ returned a message and a signature that it obtained from a query for this message, which means that $\mathcal{A}$ did not break strong unforgeability.

□

Remark 3. Sig (and thus **BSig** constructed from it in the next section) can also sign bit strings (matching thus the standard definition of digital signatures) if we assume a collision-resistant hash function $\mathcal{H}\colon \{0,1\}^* \to \mathbb{Z}_p$. Define $\mathbf{Sig}^* := (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}^*, \mathsf{Ver}^*)$ with

$$\mathsf{Sign}^*(sk, m) := \mathsf{Sign}(sk, (G^{\mathcal{H}(m)}, H^{\mathcal{H}(m)}))$$
$$\mathsf{Ver}^*(vk, \Sigma, m) := \mathsf{Ver}(vk, \Sigma, (G^{\mathcal{H}(m)}, H^{\mathcal{H}(m)}))$$

Security against chosen-message attacks follows by a straightforward reduction to security of **Sig** and collision resistance of $\mathcal{H}$.

4.2 Blind Automorphic Signatures

We now show how to combine automorphic signatures with the Groth-Sahai (GS) proof system to construct the first round-optimal blind signature scheme **BSig**, satisfying the strong security requirements defined in Sect. 2.2.4 (p. 17). Similarly to Fischlin's generic construction (which we discuss in Sect. 6.1.1, p. 66), our blind signatures are defined as a witness-indistinguishable proof of knowledge of a signature from an underlying scheme, which thus perfectly hides the signature. We furthermore have to ensure that the signer does not learn the message while signing. In our scheme the user sends a *randomization* of the message, on which the signer makes a "pre-signature". By adapting the randomness, the user can retrieve a signature *on the message*, rather than on a

Obtain$(ck, vk, (M, N))$**.** Choose $t \leftarrow \mathbb{Z}_p$, $\mu, \nu, \psi, \chi \leftarrow \mathcal{R}$, set $P := G^t$, $Q := H^t$, and send the following:

$$\begin{array}{lll} \mathbf{c}_M := \mathsf{Com}(ck, M, \mu) & \mathbf{c}_N := \mathsf{Com}(ck, N, \nu) & \pi_M := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (M, \mu), (N, \nu)\big) \\ \mathbf{c}_P := \mathsf{Com}(ck, P, \psi) & \mathbf{c}_Q := \mathsf{Com}(ck, Q, \chi) & \pi_P := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (P, \psi), (Q, \chi)\big) \\ U := T^t \cdot M & & \pi_U := \mathsf{Prove}\big(ck, \mathrm{E}_U, (M, \mu), (Q, \chi)\big) \end{array}$$

with Equation $\mathrm{E}_{\mathcal{DH}}(X, Y)$ defined as $e(G^{-1}, \underline{Y})\, e(\underline{X}, H) = 1_T$
and $\mathrm{E}_U(M, Q)$ defined as $e(T, \underline{Q})\, e(\underline{M}, H) = e(U, H)$.

Issue(ck, x)**.** Receive $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$. If all proofs are valid, choose $c, r \leftarrow \mathbb{Z}_p$ and send:

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r$$

The user sets $R := R' \cdot P$, $S := S' \cdot Q$, and checks whether (A, B, D, R, S) is valid on (M, N) under pk. If so, he chooses $\alpha, \beta, \delta, \rho, \sigma \leftarrow \mathcal{R}$, sets

$$\begin{array}{lll} \mathbf{c}_1 := \mathsf{Com}(ck, A, \alpha) & \mathbf{c}_2 := \mathsf{Com}(ck, B, \beta) & \mathbf{c}_3 := \mathsf{Com}(ck, D, \delta) \\ \mathbf{c}_4 := \mathsf{Com}(ck, R, \rho) & \mathbf{c}_5 := \mathsf{Com}(ck, S, \sigma) & \end{array}$$

$\vec{\pi} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}}, (A, \alpha), (B, \beta), (D, \delta), (R, \rho), (S, \sigma))$, with $\mathrm{E}_{\mathsf{Ver}}$ defined as the equations in (4.2), and outputs $(\vec{\mathbf{c}}, \vec{\pi})$.

Figure 4.1: Two-move blind signing protocol.

commitment to which the user has to commit again and prove knowledge of the opening, as in Fischlin's construction. This increases useability of our blind signatures for applications, such as anonymous proxy signatures, and also makes them shorter.

To obtain a blind signature on (M, N), the user randomly picks $t \leftarrow \mathbb{Z}_p$ and *blinds* M by the factor T^t. In addition to $U := T^t \cdot M$, she sends a GS proof of knowledge of (M, N, G^t, H^t). The signer now formally produces a signature[3] on U, for which we have $A = (K \cdot T^r \cdot U)^{1/(x+c)} = (K \cdot T^{r+t} \cdot M)^{1/(x+c)}$; thus A is the first component of a signature on (M, N) with randomness $r + t$. The user can complete the signature by adapting randomness r to $r + t$ in the last two signature components. The blind signature is a GS proof of knowledge of this signature.

Scheme 3 (BSig). $\mathsf{Setup}_{\mathrm{B}}(grp)$ *runs* $(G, F, K, T, H) \leftarrow \mathsf{Setup}(grp)$ *and* $ck \leftarrow \mathsf{Setup}(grp)$ *and returns these outputs as common parameters* pp*. As for* **Sig***, the message space is* $\mathcal{DH}$*.*

$\mathsf{KeyGen}_{\mathrm{B}}$ *is defined as* KeyGen*.*

$\mathsf{Issue} \leftrightarrow \mathsf{Obtain}$ *The blind signing protocol is given in Figure 4.1.*

$\mathsf{Ver}_{\mathrm{B}}(pp, (X, Y), (M, N), (\vec{\mathbf{c}}, \vec{\pi}))$ *For* $(X, Y), (M, N) \in \mathcal{DH}$, $\mathsf{Ver}_{\mathrm{B}}$ *runs* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}}, \vec{\mathbf{c}}, \vec{\pi})$*, with* $\mathrm{E}_{\mathsf{Ver}}$ *being the equations in* (4.2) *over the variables* (A, B, D, R, S)*.*

Using soundness of Groth-Sahai proofs, unforgeability is shown by reduction to unforgeability of **Sig**, which holds under ADH-SDH and AWF-CDH. In the WI setting, two GS proofs of knowledge of different signatures on the same message are indistinguishable; moreover, the issuer gets no

[3]Note that the user does *not* obtain a signature on U (unless $U = M$), since it is not an element of the message space; to produce $(U, H^{\log_G U}) \in \mathcal{DH}$, the user would have to break AWF-CDH.

information on the message during the issuing protocol. Together this implies blindness under DLIN or SXDH, depending on the employed instantiation of GS proofs.

Theorem 4. *Under Assumptions ADH-SDH and SXDH (or ASH-SDH, WF-CDH and DLIN for symmetric groups), scheme* **BSig** *is an unforgeable blind-signature scheme.*

Proof. The protocol is correct: The signer sends $A = (K \cdot T^r \cdot U)^{\frac{1}{x+c}} = (K \cdot T^{r+t} \cdot M)^{\frac{1}{x+c}}$, $B = F^c$, $D = H^c$, $R' = G^r$, $S' = H^r$ and the user sets $R := R' \cdot P = G^{r+t}$ and $S := S' \cdot Q = H^{r+t}$, which makes (A, B, D, R, S) a valid signature on (M, N). By completeness of GS proofs, the blind signature $(\vec{\mathbf{c}}, \vec{\pi})$ is accepted by $\mathsf{Ver}_{\mathrm{B}}$.

Blindness. *If we are given two messages from the adversary and run* Obtain *twice for these messages (in random order) with the adversary, and then give the two resulting signature/message pairs, then the adversary cannot relate them to their issuings.*

We modify the security game by setting $ck \leftarrow \mathsf{WISetup}$ (leading to perfectly WI commitments and proofs). This modification is indistinguishable by DLIN or SXDH (depending on the used Groth-Sahai instantiation). A signature/message pair $\big((\vec{\mathbf{c}}, \vec{\pi}), (M, N)\big)$ that the adversary gets in the end now perfectly hides the signature, since the commitments are under ck. Moreover, for every pair $(M', N') \in \mathcal{DH}$, there exists $t' \in \mathbb{Z}_p$ s.t. $U = T^{t'} \cdot M'$. By witness indistinguishability of Groth-Sahai proofs, every such tuple $(M', N', P' := G^{t'}, Q' := H^{t'})$ leads to the same distribution of $(\mathbf{c}_M, \mathbf{c}_N, \mathbf{c}_P, \mathbf{c}_Q, \pi_M, \pi_P, \pi_U)$. The adversary's view after the first round of the protocol is thus independent of (M, N).

Unforgeability. *After running the protocol $q-1$ times with an honest signer, no adversary can output q different messages and valid blind signatures on them.*

We reduce unforgeability to the security of the signature scheme **Sig**, which follows from ADH-SDH and AWF-CDH by Theorem 3 (and thus from ADH-SDH and SXDH by Lemma 1). Given parameters $pp' = (G, F, K, T, H)$ and a public key (X, Y) for **Sig**, we first run $(ck, ek) \leftarrow \mathsf{ExSetup}$ and give $pp = (pp', ck)$ to an adversary $\mathcal{A}$ against unforgeability of **BSig**. We then run the protocol (simulating the signer) with adversary $\mathcal{A}$ as follows. Whenever $\mathcal{A}$ sends a correct tuple $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$, we use ek to extract (M, N, P, Q). Soundness of the proofs π_M, π_P, π_U ensures that there exist $m, t \in \mathbb{Z}_p$ s.t. $M = G^m, N = H^m, P = G^t, Q = H^t$ and $U = T^t \cdot M$. We query our Sign oracle for a signature on (M, N). On receiving (A, B, D, R, S), we give the adversary $(A, B, D, R' := R \cdot P^{-1}, S' := S \cdot Q^{-1})$. This perfectly simulates Issue: let c and $\hat{r}$ be such that $B = F^c$ and $R = G^{\hat{r}}$; then $A = (K \cdot T^{\hat{r}} \cdot M)^{\frac{1}{x+c}} = (K \cdot T^{\hat{r}-t} \cdot U)^{\frac{1}{x+c}}$, $R' = G^{\hat{r}-t}$ and $S' = H^{\hat{r}-t}$, which corresponds to a real Issue reply using randomness c and $r := \hat{r} - t$.

The adversary wins the game if after $q-1$ issuings, it outputs q blind signatures on different messages. We extract the **Sig** signature on a message which we did not query our own oracle. By soundness of GS proofs, this is a valid signature and can thus be returned as a forgery.

□

The round complexity of the scheme is optimal [Fis06]. In the DLIN instantiation, the user sends 22 group elements (GE), since all proofs are for linear equations (which cost only 3 group elements; cf. Sect. 2.4), and the signer sends 5 GE. Blind signatures consist of 30 GE ($\vec{\mathbf{c}}$ is in $\mathbb{G}^{5\times 3}$ and $\vec{\pi}$ consists of $9 + 2 \cdot 3$ GE). In the SXDH instantiation, the user message is in $\mathbb{G}_1^{17} \times \mathbb{G}_2^{16}$, the signer message in $\mathbb{G}_1^3 \times \mathbb{G}_2^2$ and a blind signature is in $\mathbb{G}_1^{18} \times \mathbb{G}_2^{16}$. Note that the scheme remains automorphic, since commitments and proofs are composed of group elements and are verified by checking PPEs.

If we base **BSig** on a symmetric bilinear group and the scheme $\mathbf{Sig}_1$ rather than **Sig**, we obtain a round-optimal blind signature scheme which is not automorphic but which is secure under *DH-SDH*, WF-CDH and DLIN, thus under weaker assumptions, since ADH-SDH implies DH-SDH.

Remark 4 (Signing Committed Values)**.** The core building block for P-signatures [BCKL08] is an interactive protocol allowing a user that published a commitment to obtain a signature on the committed value. If the user publishes $(\mathbf{c}_M, \mathbf{c}_N, \pi_M)$ before running the blind-signature protocol we get exactly this. Note that this is due to our specific construction which differs from Fischlin's where the user only gets a signature on a *commitment* to the message. We will further discuss P-signatures in Sect. 6.1.2 (p. 67).

4.3 Automorphic Signatures on Message Vectors

In order to sign vectors of messages of arbitrary length, we proceed as follows. We first show how to transform any signature scheme whose message space $\mathcal{M}$ forms an algebraic group (and contains the public-key space) into one that signs 2 messages at once—if we exclude the neutral element from the message space of the transform. The message space will thus be $\mathcal{M}^* \times \mathcal{M}^*$ with $\mathcal{M}^* := \mathcal{M} \setminus \{1\}$. A signature on a message pair will contain 3 signatures (of the original scheme) on different *linear combinations* of the components. At the end of the section we show that 3 are indeed necessary. Note that $\mathcal{DH}$, the message space for the schemes $\mathbf{Sig}_1$ and **Sig**, is a group when the group operation is defined as component-wise multiplication.

We then give a straightforward generic transformation from any scheme signing 2 messages (and whose verification keys lie in the message space) to one signing message vectors of arbitrary length (Definition 3). Both transformations do not modify setup and key generation and they are invariant w.r.t. the structure of verification; in particular, if the verification predicate of the original scheme is a conjunction of PPEs then so is that of the transform.

Definition 2 (Pair transform)**.** *Let* $\mathbf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme whose message space* $(\mathcal{M}, \cdot)$ *is an algebraic group that contains the verification keys. The* pair transform *of* **Sig** *with message space* $\mathcal{M}^* \times \mathcal{M}^*$ *is defined as* $\mathbf{Sig}' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}', \mathsf{Ver}')$ *with*

$\mathsf{Sign}'(sk, (M_1, M_2))$**:** *Set* $(vk_0, sk_0) \leftarrow \mathsf{KeyGen}$ *and return*

$$\Sigma := \big(vk_0,\ \mathsf{Sign}(sk, vk_0),\ \mathsf{Sign}(sk_0, M_1),\ \mathsf{Sign}(sk_0, M_1 \cdot M_2),\ \mathsf{Sign}(sk_0, M_1 \cdot M_2^3)\big)\ .$$

$\mathsf{Ver}'\big(vk, (M_1, M_2), (vk_0, \Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)\big)$**:** *Return 1 if all of the following are 1:*

$$\mathsf{Ver}(vk, vk_0, \Sigma_0) \quad \mathsf{Ver}(vk_0, M_1, \Sigma_1) \quad \mathsf{Ver}(vk_0, M_1 \cdot M_2, \Sigma_2) \quad \mathsf{Ver}(vk_0, M_1 \cdot M_2^3, \Sigma_3)$$

Theorem 5. *If* **Sig** *is EUF-CMA secure then so is* $\mathbf{Sig}'$*.*

Proof. Consider an adversary $\mathcal{A}$ making q queries on message pairs $(M_1^{(i)}, M_2^{(i)})$ for $1 \le i \le q$ and outputting a new message (M_1^*, M_2^*) and a valid signature $\Sigma^* = (vk_0^*, \Sigma_0^*, \Sigma_1^*, \Sigma_2^*, \Sigma_3^*)$ on it. Let vk be a challenge for **Sig**. We call adversaries Type 1 if $vk_0^* \neq vk_0^{(i)}$ for all $1 \le i \le q$. Type 1 forgeries are reduced by giving vk to the adversary as the challenge key and answering signing queries by choosing $(vk_0, sk_0) \leftarrow \mathsf{KeyGen}$, querying vk_0 to the signing oracle and using sk_0 to complete a $\mathbf{Sig}'$ signature. From the adversary's output we can return (vk_0^*, Σ_0^*) as a forgery under vk.

Forgeries of Type 2, i.e., where for some i we have $vk_0^* = vk_0^{(i)}$, are handled as follows. Let vk be a **Sig** challenge key. We choose $(vk', sk') \leftarrow \mathsf{KeyGen}$ and $i^* \leftarrow \{1, \ldots, q\}$ and give vk' to the adversary. Knowing sk', we answer the signing queries by running $\mathsf{Sign}'(sk', \cdot)$—except for the i^*-th query: being queried message (M_1, M_2), we set $vk_0^{(i^*)} := vk$, and use our signing oracle on messages M_1, $M_1 \cdot M_2$ and $M_1 \cdot M_2^3$ to simulate a $\mathbf{Sig}'$ signature. We show that if we guessed correctly ($i^* = i$) then from $\mathcal{A}$'s output we can extract a forgery under vk.

In particular, we show that any valid forgery Σ^* with $vk_0^* = vk$ on (M_1^*, M_2^*) must contain a signature on a message we have not queried to our oracle. We proceed by case distinction: if Σ_1^*, the signature on M_1^*, is a signature on a message we have queried our oracle then M_1^* is either M_1, $M_1 \cdot M_2$ or $M_1 \cdot M_3^3$.

- $M_1^* = M_1$. In this case, if the message of Σ_2^* (i.e., $M_1^* \cdot M_2^*$) has also been queried, then either
  - $M_1^* \cdot M_2^* = M_1$, thus $M_2^* = 1$ which is not in the message space and thus the adversary did not win, or
  - $M_1^* \cdot M_2^* = M_1 \cdot M_2$, thus $M_2^* = M_2$, thus the adversary did not return a valid forgery since $(M_1^*, M_2^*) = (M_1, M_2)$, or
  - $M_1^* \cdot M_2^* = M_1 \cdot M_2^3$, thus $M_2^* = M_2^3$, thus Σ_3^* is a valid signature on $M_1^* \cdot (M_2^*)^3 = M_1 \cdot M_2^9$, which we have not queried to our oracle, since $M_2 \neq 1$ (see below).
- $M_1^* = M_1 \cdot M_2$. Again, if we queried $M_1^* \cdot M_2^*$, then either
  - $M_1^* \cdot M_2^* = M_1$, thus $M_2^* = M_2^{-1}$, thus Σ_3^* is a valid signature on $M_1^* \cdot (M_2^*)^3 = M_1 \cdot M_2^{-2}$, which we have not queried to our oracle, or
  - $M_1^* \cdot M_2^* = M_1 \cdot M_2$, thus $M_2^* = 1$, which is not a valid message, or
  - $M_1^* \cdot M_2^* = M_1 \cdot M_2^3$, thus $M_2^* = M_2^2$, thus Σ_3^* is a valid signature on $M_1^* \cdot (M_2^*)^3 = M_1 \cdot M_2^7$, which we have not queried to our oracle.
- $M_1^* = M_1 \cdot M_2^3$. Again, if we queried $M_1^* \cdot M_2^*$, then either
  - $M_1^* \cdot M_2^* = M_1$, thus $M_2^* = M_2^{-3}$, thus Σ_3^* is a valid signature on $M_1^* \cdot (M_2^*)^3 = M_1 \cdot M_2^{-6}$, which we have not queried to our oracle, or
  - $M_1^* \cdot M_2^* = M_1 \cdot M_2$, thus $M_2^* = M_2^{-2}$, thus Σ_3^* is a valid signature on $M_1^* \cdot (M_2^*)^3 = M_1 \cdot M_2^{-3}$, which we have not queried to our oracle, or
  - $M_1^* \cdot M_2^* = M_1 \cdot M_2^3$, thus $M_2^* = 1$, which is not a valid message.

Note that all the above messages were indeed not queried to the oracle: they are all of the form $M_1 \cdot M_2^i$ with $i \notin \{0, 1, 3\}$, whereas the messages queried to the **Sig** oracle are of the form $M_1 \cdot M_2^j$ with $j \in \{0, 1, 3\}$. If we had $M_1 \cdot M_2^i = M_1 \cdot M_2^j$ for any of the above values of i and j, we would have $M_2^{i-j} = 1$ for $i \neq j$ and thus $M_2 = 1$, which would not have been accepted in a signing request. We thus showed that any valid message/signature pair the adversary returns contains a forgery. □

Definition 3 (Vector transform)**.** *Let* $\mathbf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ *be a signature scheme with message space* $\mathcal{M} \times \mathcal{M}$*, such that* $\mathcal{M}$ *contains the verification keys. If there exists an efficiently computable injection* $\mathsf{Inj}\colon \{1, \ldots, |\mathcal{M}|\} \to \mathcal{M}$ *then the* vector transform *of* **Sig** *is defined as* $\mathbf{Sig}'' = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}'', \mathsf{Ver}'')$ *with*

$\mathsf{Sign}''(sk, (M_1, \ldots, M_n))$**:** *Set* $(vk_0, sk_0) \leftarrow \mathsf{KeyGen}$ *and return*

$$\Sigma := \big(vk_0,\, \mathsf{Sign}(sk, vk_0, \mathsf{Inj}(n)), \mathsf{Sign}(sk_0, M_1, \mathsf{Inj}(1)), \ldots, \mathsf{Sign}(sk_0, M_n, \mathsf{Inj}(n))\big)\ .$$

$\mathsf{Ver}''\big(vk, (M_1, \ldots, M_n), (vk_0, \Sigma_0, \Sigma_1, \ldots, \Sigma_n)\big)$**:** *Return 1 if the following are 1:*

$$\mathsf{Ver}\big(vk, (vk_0, \mathsf{Inj}(n)), \Sigma_0\big) \qquad \text{For all } i \in \{1, \ldots, n\} : \ \mathsf{Ver}\big(vk_0, (M_i, \mathsf{Inj}(i)), \Sigma_i\big)$$

Theorem 6. *If* **Sig** *is EUF-CMA secure then so is* **Sig**$''$*.*

Proof. Let q be the maximal number of the adversary's signing queries. Let $\vec{M}^{(i)} := (M_1^{(i)}, \ldots, M_{n_i}^{(i)})$ denote the adversary's i-th signing query, let $\Sigma^{(i)} := (vk_0^{(i)}, \Sigma_0^{(i)}, \ldots, \Sigma_{n_i}^{(i)})$ denote the responses, and let the adversary's final output be $\big((M_1^*, \ldots, M_{n^*}^*), \Sigma^* = (vk_0^*, \Sigma_0^*, \ldots, \Sigma_{n^*}^*)\big)$. Let vk be a challenge for **Sig**. We distinguish two types of forgers and show how to reduce them to EUF-CMA of **Sig**.

1. $\forall i : \big(vk_0^* \neq vk_0^{(i)} \vee n^* \neq n_i\big)$. Give vk to the adversary and answer the i-th signing query by setting $(vk_0^{(i)}, sk_0^{(i)}) \leftarrow \mathsf{KeyGen}$, querying $(vk_0^{(i)}, \mathsf{Inj}(n_i))$ to the Sign-oracle and using $sk_0^{(i)}$ to sign $(M_j^{(i)}, \mathsf{Inj}(j))$ for all j. If Σ^* is of Type 1, then $((vk_0^*, \mathsf{Inj}(n^*)), \Sigma_0^*)$ is a forgery under vk.
2. $\exists i : \big(vk_0^* = vk_0^{(i)} \wedge n^* = n_i\big)$. Choose $i^* \leftarrow \{1, \ldots, q\}$, produce $(vk', sk') \leftarrow \mathsf{KeyGen}$ and give the adversary vk' as challenge. Answer all queries as in the protocol, except for the i^*-th query: set $vk_0^{(i^*)} := vk$ and query signatures on $(M_j^{(i^*)}, \mathsf{Inj}(j))$ for all $1 \leq j \leq n^*$ to the Sign oracle and complete the signature using sk'. Suppose Σ^* is of Type 2 and we guessed correctly ($i^* = i$). Since $(M_1^*, \ldots, M_{n_i}^*)$ is a valid forgery, for some $1 \leq j \leq n_i$ we have $M_j^* \neq M_j^{(i)}$. Thus $\big((M_j^*, \mathsf{Inj}(j)), \Sigma_j^*\big)$ is a valid forgery under vk for a message we did not query.

□

We now discuss why the construction in Definition 2 is optimal and why it seems hard to construct a vector transform directly.

A Discussion on the Vector Transformation. Transforming a signature scheme whose verification keys lie in the message space to one that signs vectors of messages of arbitrary length and leaving the structure of verification invariant is somewhat hard. The standard approach to sign long messages is to hash them and sign the hash. But the resulting scheme would not be automorphic, so we have to find a different strategy. An idea that comes to mind is the following: For each signature, the signer first produces a temporary key pair (vk, sk), signs vk with her secret key and uses sk to sign every component of the vector. An easy attack would be to reorder the messages of a queried vector. To prevent this shuffling attack, we could let sk sign one transient key per message component, which will sign the message and its *index*. To thwart an attack that returns a truncated message, we also sign the length.

To sign the indices and the length, we need to assume an injection Inj from natural numbers into the message space as in Definition 3. The above construction however succumbs to a series of attacks, which come from the fact that verification keys, images under Inj, and message all have the same form, which is inherent. An adversary could for example query a signature on the message $(\mathsf{Inj}(2), \mathsf{Inj}(1))$ and return a signature on $(\mathsf{Inj}(1), \mathsf{Inj}(2))$ by simply reordering the signature components. If however we start from a signature scheme signing 2 messages, we avoid all these problems as can be seen by the natural construction in Definition 3 and the straightforward proof of Theorem 6.

The crucial step is thus that from 1 to 2 messages. If we assume some structure on the message space (which is the case for our constructions, since messages are elements of an algebraic group), then we could try to sign several messages at once by signing their *product*. Again, we first sign a "one-time" key with the actual key, and use that key to produce the signatures contained in a

signature of the transform. This prevents the adversary from combining signatures received from different queries and we therefore effectively only have to handle one-time attacks. As it turns out, we need to construct the messages we actually sign very carefully to prevent the adversary from deriving a signature on a new message from a signing-query response. If we only sign *one* product of the components, there are trivial attacks. Signing two products seems more promising, but we show that this do not suffice either:

Concretely, we want to devise a one-time scheme that signs (M_1, M_2) by signing two linear combinations of the messages; i.e., a signature on (M_1, M_2) consists of a signature on $(M_1^{a_1} \cdot M_2^{a_2})$ and one on $(M_1^{b_1} \cdot M_2^{b_2})$, for some fixed $(a_1, a_2, b_1, b_2) \in \mathbb{Z}^4$.

Assume first that (a_1, a_2) and (b_1, b_2) are linearly dependent, i.e., $b_1 = ca_1$ and $b_2 = ca_2$ for some c and that $a_1 \neq 0$ (otherwise signatures would be independent of M_1 and thus easily forgeable). After querying a transform signature on (M_1, M_2) (and thus receiving signatures on $(M_1^{a_1} \cdot M_2^{a_2})$ and $(M_1^{ca_1} \cdot M_2^{ca_2})$), one can produce a forgery as follows: set $M_1^* := M_1 \cdot M_2^{a_2/a_1}(M_2^*)^{-a_2/a_1}$ for an arbitrary $M_2^* \neq M_2$. A signature on this message consists thus of a signature on $(M_1^*)^{a_1} \cdot (M_2^*)^{a_2} = M_1^{a_1} \cdot M_2^{a_2}$ and $(M_1^*)^{ca_1} \cdot (M_2^*)^{ca_2} = M_1^{ca_1} \cdot M_2^{ca_2}$, thus the precise two messages for which we have signatures from the signing query.

Assume now that (a_1, a_2) and (b_1, b_2) are linearly independent, i.e., $a_1b_2 - b_1a_2 \neq 0$; w.l.o.g., assume that $b_2 \neq 0$. Querying (M_1, M_2) yields signatures Σ_1 and Σ_2 on $(M_1^{a_1} \cdot M_2^{a_2})$ and $(M_1^{b_1} \cdot M_2^{b_2})$, respectively. Setting $M_1^* := M_1^{(b_1b_2 - a_1a_2)/D} \cdot M_2^{(b_2^2 - a_2^2)/D}$ (with $D := a_1b_2 - b_1a_2$) and $M_2^* := M_1^{a_1/b_2} \cdot M_2^{a_2/b_2} \cdot (M_1^*)^{-b_1/b_2}$ makes $(M_1^*)^{a_1} \cdot (M_2^*)^{a_2} = M_1^{b_1} \cdot M_2^{b_2}$ and $(M_1^*)^{b_1} \cdot (M_2^*)^{b_2} = M_1^{a_1} \cdot M_2^{a_2}$, thus we can reuse the signatures, i.e., produce a forgery (Σ_2, Σ_1) on (M_1^*, M_2^*). We have thus shown that against any scheme signing 2 linear combinations, there are attacks.

Moreover, note that finding *three* linear combinations leading to a valid scheme is not trivial either. E.g., choosing M_1, $M_1 \cdot M_2$ and $M_1 \cdot M_2^2$ succumbs to the following attack: Setting $M_1^* := M_1 \cdot M_2^2$ and $M_2^* := M_2^{-1}$, we can recycle and reorder the signatures from the query.

A Direct Triple Transform. As an additional result we give a transformation from a scheme signing single messages to one signing *triples* of messages. Again we have to exclude the neutral element, thus if the message space of the original scheme is $\mathcal{M}$ then that of the transform is $\mathcal{M}^* \times \mathcal{M}^* \times \mathcal{M}^*$. Rather than via the detour of the pair and the vector transform, we give a direct construction: To sign a message (M_1, M_2, M_3), choose a transient key pair, sign the verification key, and use the signing key to sign the following 7 linear combinations:

$$M_1 \cdot M_2 \qquad M_1 \cdot M_3 \qquad M_1 \cdot M_2^3 \cdot M_3^5$$
$$M_1^{-1} \cdot M_2^8 \cdot M_3^{17} \qquad M_1^2 \cdot M_2^5 \cdot M_3^{16} \qquad M_1^7 \cdot M_2^4 \cdot M_3^{14} \qquad M_2^{13} \cdot M_3^{21}$$

Whereas in the proof of Theorem 5 we had $3^3 = 27$ cases to distinguish, for our triple transform there are $7^7 = 823\,543$ cases. We were able to give a computer-aided proof of unforgeability of the triple transform.

Chapter 5

Commuting Signatures and Verifiable Encryption

Contents

In this chapter we formally define the notion of *commuting signatures and verifiable encryption*. To instantiate the concept we identify several properties of Groth-Sahai proofs (Sect. 5.3). The results (as well as those of Chapter 7) appear in [8].

5.1 Verifiably Encrypted Signatures

Consider an extractable commitment scheme **Com** (where committed values can be extracted using the extraction key), a proof system **Proof** for **Com**, which allows to prove that committed values satisfy an equation, and a *compatible* signature scheme **Sig**, that is, whose verification keys, messages and signatures can be committed to via **Com**, and validity of a committed signature can be proven with **Proof**. (see Sect. 2.2, p. 14, for the formal definitions). From the triple $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ one can now derive a *verifiable encryption scheme* satisfying the definitions of Rückert and Schröder [RS09], who revisited those of Boneh et al. [BGLS03]—if a proof that a committed signature is valid on a message M under a key vk (i.e., it satisfies $\mathsf{Ver}(vk, M, \cdot)$) can

be *simulated* (see Sect. 7.4). This is the case for our instantiations[1], given in Sections 2.4.2, 2.4.3 and 4.1.

A scheme for verifiably encrypted signatures allows a signer to encrypt a signature under a trusted third party's key and give a proof that the plaintext is a valid signature. The classical application is *contract signing* between two parties where each one wants to ensure that the other party signs as well. Both parties first produce a verifiably encrypted signature on the contract, send it to each other, and then reveal the signature. In case one party refuses, the other one can call the "adjudicator" who holds the decryption key to retrieve the signature.

Definition 4 (Verifiably encrypted signatures (VES))**.** *A verifiably encrypted signature scheme is defined as the tuple* $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf})$. Kg *outputs a signature key pair* (vk, sk), Sig *and* Vf *produce and verify signatures.* AdjKg *outputs a key pair* (apk, ask) *for the* adjudicator. $\mathsf{Create}(sk, apk, M)$ *returns a VES* Ω *which is verified by* $\mathsf{VesVf}(apk, vk, \Omega, M)$. *Given a VES,* $\mathsf{Adj}(ask, apk, vk, \Omega, M)$ *returns a signature* Σ *on* M *under* vk. *The security notions from* [RS09] *are the following:*

- Unforgeability *means that no adversary given the public keys and access to a* Create *and an* Adj *oracle can output a VES on a message* M *that it has never queried to its oracles.*
- Abuse freeness *states that no malicious adjudicator provided with a* Create *oracle can output a valid VES for a message it never queried.*
- Extractability *means that no malicious signer that can create its own* vk *and has access to an* Adj *oracle can output a valid VES from which cannot be extracted a valid signature.*
- Opacity *means that no adversary given* vk *and* apk *and access to oracles* Create *and* Adj *for messages of its choice, can output a valid pair* (M, Σ) *if it has never queried* Adj *on* M.

A Straightforward Instantiation. Based on $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ we might instantiate a VES scheme as follows. The adjudicator's key generation AdjKg runs $\mathsf{Setup}_{\mathrm{S}}$ and $(ck, ek) \leftarrow \mathsf{ExSetup}$, and defines apk as the signature parameters and ck, and sets $ask := ek$. The signer's key generation, signing and verification are defined as $\mathsf{Kg} := \mathsf{KeyGen}_{\mathrm{S}}$, $\mathsf{Sig} := \mathsf{Sign}$, and $\mathsf{Vf} := \mathsf{Ver}$. $\mathsf{Create}(sk, ck, M)$ creates a VES on M by setting $\Sigma \leftarrow \mathsf{Sign}(sk, M)$, choosing $\rho \leftarrow \mathcal{R}$ and returning $\Omega := \big(\mathbf{c}_\Sigma = \mathsf{Com}(ck, \Sigma, \rho),\ \widetilde{\pi} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk, M, \cdot)}, (\Sigma, \rho))\big)$. Verification $\mathsf{VesVf}(ck, vk, (\mathbf{c}_\Sigma, \widetilde{\pi}), M)$ is defined as $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk, M, \cdot)}, \mathbf{c}_\Sigma, \widetilde{\pi})$. Finally, $\mathsf{Adj}(ek, ck, vk, \Omega, M)$ checks whether $\Omega = (\mathbf{c}_\Sigma, \widetilde{\pi})$ is valid and if so returns $\mathsf{Extr}(ek, \mathbf{c}_\Sigma)$.

The scheme $(\mathsf{Kg}, \mathsf{AdjKg}, \mathsf{Sig}, \mathsf{Vf}, \mathsf{Create}, \mathsf{VesVf}, \mathsf{Adj})$ almost satisfies the security notions from Definition 4. The first three notions are reduced to unforgeability of **Sig** and soundness of **Proof** in a straightforward manner. Note in particular that **Proof** is *perfectly* sound, so no adversary even when given the extraction key can make a proof of a false statement (as required by abuse freeness).

Opacity requires to simulate proofs since an adversary can query Create for a message M and output a signature on M—which is considered successful as long as it did not query Adj. This notion can be proved by a reduction with a security loss that is exponential in the number of Adj calls:[2] Let Game 0 be the original game. In Game 1 we abort if the adversary makes an Adj query for a message it never queried Create for; or if it makes an Adj query for $(\mathbf{c}_\Sigma, \widetilde{\pi})$ such that the committed value Σ was never used to answer one of the Create queries. By strong unforgeability

[1] Groth-Sahai proofs for pairing-product equations can be simulated if the equations only contain elements from $\mathbb{G}_1$ and $\mathbb{G}_2$ but not from $\mathbb{G}_T$. This is satisfied by the equations in (4.2) in Sect. 4.1, which constitute the verification equations for which we make proofs in our instantiation. See Sect. 7.4 (p. 85) for how to simulate such proofs.

[2] If the adversary is only allowed a *constant* number of Adj queries, this is sufficient (see also Remark 5 below).

of **Sig** and soundness of **Proof**, the probability of aborting is negligible. In Game 2, when queried $\mathcal{O}_{\mathsf{Adj}(ek,ck,vk,\cdot,\cdot)}((\mathbf{c}_\Sigma, \widetilde{\pi}), M)$, instead of extracting the signature committed to in $\mathbf{c}_\Sigma$, we return the signature produced when answering $\mathcal{O}_{\mathsf{Create}(sk,ck,\cdot)}(M)$; if there have been more such calls then we guess randomly. (If, in the worst case, the adversary queries Create q_C times and Adj q_A times on the *same* message then the probability of correctly simulating Game 2 is $1/q_C^{q_A}$.) Game 3 is Game 2 but replacing ck by ck^* output by $\mathsf{WISetup}$. Game 3 can now be simulated by a challenger playing the unforgeability game against **Sig**: Given the trapdoor for Groth-Sahai proofs in the WI setting, the challenger *simulates* the commitments and proofs to answer Create queries, i.e., without using signatures. When queried Adj on a message M, it asks its own Sign oracle for a signature on M and returns it (or returns a signature it had already returned, depending on the guess, which is made as in Game 2). A successful adversary outputs a signature on M for which it has never queried Adj (and thus never made the challenger query Sign for it) and can therefore be used to break strong unforgeability of **Sig**.

A Fully Secure Instantiation. The security reduction for opacity can be made tight if outputs of Create are *non-malleable* (i.e., from a VES Ω returned by Create on M, one cannot produce a different valid Ω' for M): the adversary can then make Adj queries only for VES received from a Create query, and in the reduction the challenger need not guess the correct signature to answer Adj queries. Non-malleability is easily achieved by replacing Ω by $(\Omega, \mathsf{Sign}(sk, \Omega))$—possibly hashing Ω to the message space first—in the definition of Create and adding a check of the second component to VesVf. Non-malleability then follows from strong unforgeability of **Sig**.

Remark 5. In our application to delegatable credentials, we require somewhat different properties from the triple $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$. On the one hand, we do not require opacity, since no adversary can query *extraction* of committed values—the exponential reduction of the straightforward instantiation is thus irrelevant.

On the other hand, we require that the verification keys are in $\mathcal{M}$ (i.e., **Sig** is automorphic) and that we can commit to verification keys and messages (for which we will introduce a commitment scheme $\mathbf{Com}_{\mathcal{M}}$) and prove validity of an encrypted signature, possibly on an encrypted message or under an encrypted key. We moreover require that two verifiably encrypted signatures are *indistinguishable*, i.e., it is hard to decide whether two VES encrypt the same signature. This property is implied by witness-indistinguishability of **Proof**, and is not required for VES.

5.2 Definition of Commuting Signatures

Recall the primitives introduced in Sect. 2.2. Let

$$\mathbf{Com} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{RdCom}, \mathsf{ExSetup}, \mathsf{Extr}, \mathsf{WISetup})$$

be an extractable commitment scheme with value space $\mathcal{V}$ and randomness space $\mathcal{R}$; let

$$\mathbf{Proof} = (\mathsf{Prove}, \mathsf{Verify}, \mathsf{RdProof})$$

be a randomizable WI proof system for **Com** for a class $\mathcal{E}$; and let

$$\mathbf{Sig} = (\mathsf{Setup}_{\mathrm{S}}, \mathsf{KeyGen}_{\mathrm{S}}, \mathsf{Sign}, \mathsf{Ver})$$

be a strongly unforgeable signature scheme with message space $\mathcal{M}$ that is *compatible* with **Com** and **Proof**, i.e., the components of verification keys, messages and signatures lie in $\mathcal{V}$ and the verification equations lie in $\mathcal{E}$. We extend $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig})$ by the following functionalities presented in the introduction (p. 9; see also Figure 1.1) and formally defined in Definition 5 below:

- $\mathbf{Com}_{\mathcal{M}}$ is a randomizable extractable commitment scheme with the same commitment keys as **Com** and whose message space is that of **Sig**.
- AdPrC and $\mathsf{AdPrC}_{\mathcal{M}}$ are algorithms that, given a message/signature pair of which one is committed and proven valid, produce a proof of validity when both are committed.
- AdPrDC and $\mathsf{AdPrDC}_{\mathcal{M}}$ are algorithms that, given a committed message/signature pair and a proof, as well as the randomness for one of the commitments, return an adapted proof; in particular, AdPrDC returns a proof that a signature is valid on a committed message and $\mathsf{AdPrDC}_{\mathcal{M}}$ returns a proof that a committed signature is valid on a given message.
- SigCom takes a $\mathbf{Com}_{\mathcal{M}}$ commitment and a signing key, and produces a committed signature on the committed message and a proof of validity. $\mathsf{SmSigCom}$ simulates SigCom and is given a signature instead of the signing key.
- $\mathsf{AdPrC}_{\mathcal{K}}$ is given a proof of validity for a committed signature and a committed message and adapts it to a proof for when the verification key is also committed. $\mathsf{AdPrDC}_{\mathcal{K}}$, given the randomness of the committed verification key adapts a proof to when the key is given in the clear.

AdPrC denotes thus *adapting a proof when committing* (to a signature) and AdPrDC when *decommitting*. A subscript $\mathcal{M}$ denotes proof adaption when (de)committing to a message and $\mathcal{K}$ when (de)committing to a verification key. SigCom denotes *signing committed values* and $\mathsf{SmSigCom}$ simulates SigCom.

Definition 5. *A system of* **commuting signatures and verifiable encryption** *consists of an extractable commitment scheme* **Com***, a (randomizable) WI proof system* **Proof** *for* **Com***, a compatible signature scheme* **Sig** *and the functionalities* $\mathsf{Com}_{\mathcal{M}}, \mathsf{RdCom}_{\mathcal{M}}, \mathsf{Extr}_{\mathcal{M}}, \mathsf{AdPrC}, \mathsf{AdPrDC}, \mathsf{AdPrC}_{\mathcal{M}}, \mathsf{AdPrDC}_{\mathcal{M}}, \mathsf{AdPrC}_{\mathcal{K}}, \mathsf{AdPrDC}_{\mathcal{K}}, \mathsf{SigCom}$*, and* $\mathsf{SmSigCom}$ *defined below.*

$\mathbf{Com}_{\mathcal{M}}$. *On input* $pp = (ck, pp_{\mathrm{S}})$ *returned by* Setup *and* $\mathsf{Setup}_{\mathrm{S}}$*, respectively, a message* $M \in \mathcal{M}$ *and* $\mu \in \mathcal{R}_{\mathcal{M}}$*, algorithm* $\mathsf{Com}_{\mathcal{M}}$ *outputs a commitment* $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$*, the space of commitments.* $\mathsf{RdCom}_{\mathcal{M}}$ *takes inputs* pp*,* $\mathbf{C}$ *and* $\mu' \leftarrow \mathcal{R}_{\mathcal{M}}$ *and outputs a randomized commitment* $\mathbf{C}'$*. On input* ek *output by* $\mathsf{ExSetup}$*, and* $\mathbf{C}$*,* $\mathsf{Extr}_{\mathcal{M}}$ *outputs the committed value* M*.*

We require that $\mathbf{Com}_{\mathcal{M}} := (\mathsf{Setup}, \mathsf{Com}_{\mathcal{M}}, \mathsf{RdCom}_{\mathcal{M}}, \mathsf{ExSetup}, \mathsf{Extr}_{\mathcal{M}}, \mathsf{WISetup})$ *is a randomizable extractable commitment scheme that is perfectly binding and computationally hiding as defined in Sect. 2.2.1. Moreover, it must be* compatible *with* **Proof***, i.e., the components of* $M \in \mathcal{M}$ *are contained in* $\mathcal{V}$*,* Prove *and* $\mathsf{RdProof}$ *accept inputs from* $\mathcal{R}_{\mathcal{M}}$ *and* Verify *accepts* $\mathbf{Com}_{\mathcal{M}}$ *commitments as inputs.*

In the following we assume that $ck \leftarrow \mathsf{Setup}$*,* $pp_{\mathrm{S}} \leftarrow \mathsf{Setup}_{\mathrm{S}}$*,* $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}(pp_{\mathrm{S}})$*,* $M \in \mathcal{M}$ *and* $\mu \in \mathcal{R}_{\mathcal{M}}$*. We let* $pp := (ck, pp_{\mathrm{S}})$*.*

$\mathsf{AdPrC}(pp, vk, \mathbf{C}, (\Sigma, \rho), \bar{\pi})$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, \mathbf{C}, \bar{\pi}) = 1$ *then the algorithm outputs* π *that is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right] \ ,$$

where M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$*.*

$\mathsf{AdPrDC}(pp, vk, \mathbf{C}, (\Sigma, \rho), \pi)$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathsf{Com}(ck, \Sigma, \rho), \pi) = 1$*, the algorithm outputs* $\bar{\pi}$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}, (M, \mu))\right] \ ,$$

where M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$*.*

$\mathsf{AdPrC}_{\mathcal{M}}(pp, vk, (M, \mu), \mathbf{c}_\Sigma, \widetilde{\pi})$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_\Sigma, \widetilde{\pi}) = 1$ *then it outputs* π *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right] \ ,$$

where Σ *and* ρ *are such that* $\mathbf{c}_\Sigma = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{AdPrDC}_{\mathcal{M}}(pp, vk, (M, \mu), \mathbf{c}_\Sigma, \pi)$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathsf{Com}_{\mathcal{M}}(pp, M, \mu), \mathbf{c}_\Sigma, \pi) = 1$, *the algorithm outputs* $\widetilde{\pi}$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, (\Sigma, \rho))\right] \ ,$$

where Σ *and* ρ *are such that* $\mathbf{c}_\Sigma = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{AdPrC}_{\mathcal{K}}(pp, (vk, \xi), \mathbf{C}, \mathbf{c}_\Sigma, \pi)$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}, \mathbf{c}_\Sigma, \pi) = 1$, *the algorithm outputs* $\widehat{\pi}$ *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, (vk, \xi), (M, \mu), (\Sigma, \rho)))\right] \ ,$$

where M, μ, Σ *and* ρ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ *and* $\mathbf{c}_\Sigma = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{AdPrDC}_{\mathcal{K}}(pp, (vk, \xi), \mathbf{C}, \mathbf{c}_\Sigma, \widehat{\pi})$. *If* $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}, \mathsf{Com}(ck, vk, \xi), \mathbf{C}, \mathbf{c}_\Sigma, \widehat{\pi}) = 1$, *the algorithm outputs* π *which is distributed as*

$$\left[\mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho)))\right] \ ,$$

where M, μ, Σ *and* ρ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ *and* $\mathbf{c}_\Sigma = \mathsf{Com}(ck, \Sigma, \rho)$.

$\mathsf{SigCom}(pp, sk, \mathbf{C})$. *If* $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$ *then the algorithm outputs a commitment to a signature and a proof of validity* $(\mathbf{c}_\Sigma, \pi)$ *which is distributed as*

$$\left[\Sigma \leftarrow \mathsf{Sign}(sk, M);\ \rho \leftarrow \mathcal{R}\ :\ \left(\mathsf{Com}(ck, \Sigma, \rho),\ \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right)\right] \ ,$$

where M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$.

$\mathsf{SmSigCom}(pp, ek, vk, \mathbf{C}, \Sigma)$. *Assume* $(ck, ek) \leftarrow \mathsf{ExSetup}$. *If* $\mathsf{Ver}(vk, \mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}), \Sigma) = 1$ *then the algorithm outputs* $(\mathbf{c}_\Sigma, \pi)$ *which is distributed as*

$$\left[\rho \leftarrow \mathcal{R}\ :\ \left(\mathsf{Com}(ck, \Sigma, \rho),\ \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, (M, \mu), (\Sigma, \rho))\right)\right] \ ,$$

where M *and* μ *are such that* $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$. [3]

We denote the algorithms of the system that extend $\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}$ *and* $\mathbf{Com}_{\mathcal{M}}$ *by*

$$\mathbf{Algs} := (\mathsf{AdPrC}, \mathsf{AdPrDC}, \mathsf{AdPrC}_{\mathcal{M}}, \mathsf{AdPrDC}_{\mathcal{M}}, \mathsf{AdPrC}_{\mathcal{K}}, \mathsf{AdPrDC}_{\mathcal{K}}, \mathsf{SigCom}, \mathsf{SmSigCom}) \ .$$

Remark 6. When we verify a signature Σ on a message M running $\mathsf{Ver}(vk, M, \Sigma)$, we implicitly assume that Ver also checks whether $M \in \mathcal{M}$. Analogously, we assume that when verifying a proof of validity by running Verify on $\mathrm{E}_{\mathsf{Ver}}$ and $\mathbf{C}$, it checks whether $\mathbf{C} \in \mathcal{C}_{\mathcal{M}}$, too.

Definition 5 implies that running $\mathsf{Com}_{\mathcal{M}}$ on M and then SigCom yields the same output (meaning the output is distributed identically) as running $\Sigma \leftarrow \mathsf{Sign}(sk, M)$ and then $\mathsf{Com}_{\mathcal{M}}$ on M, Com on Σ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$; or running Sign, then $\mathsf{Com}_{\mathcal{M}}$ on M and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$, and then Com on Σ and AdPrC; or running Sign, then Com on Σ and Prove for $\mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}$, and then $\mathsf{Com}_{\mathcal{M}}$ on M and $\mathsf{AdPrC}_{\mathcal{M}}$. And similar statements hold for sequences of algorithm executions including decommitments and proof adaptation. This means that the diagram in Figure 1.1 (p. 9) *commutes*.

[3] Note that $\mathsf{SmSigCom}$ is not trivial: given ek it might recover the message M but not the randomness μ used for $\mathbf{C}$. The difference to AdPrC is that $\mathsf{SmSigCom}$ does not get $\bar{\pi}$ as input but ek instead.

5.2.1 Black-Box Results

We immediately get the following properties of commuting signatures, which follow from the security of the used building blocks and the fact that all algorithms perfectly commute.

Unforgeability. Extractability of **Com** and $\mathbf{Com}_{\mathcal{M}}$, perfect soundness of **Proof**, strong unforgeability of **Sig** and commutativity of SigCom with $\mathsf{SmSigCom}$ and signing and verifiably encrypting implies the following notion unforgeability, defined as the intractability for a p.p.t. adversary $\mathcal{A}$ of winning the following game:

> Run $(ck, ek) \leftarrow \mathsf{ExSetup}$ and $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}(\mathsf{Setup}_{\mathrm{S}})$; provide $\mathcal{A}$ with (ck, ek, vk) and access to a SigCom oracle that on input $\mathbf{C}$, a commitment to a message, outputs $\mathsf{SigCom}(ck, sk, \mathbf{C})$. Let $\mathbf{C}_i$ be the value submitted in the i-th oracle call and $(\mathbf{c}_i, \pi_i)$ be the response; define $M_i := \mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}_i)$ and $\Sigma_i := \mathsf{Extr}(ek, \mathbf{c}_i)$. Then $\mathcal{A}$ wins if it outputs $(\mathbf{C}^*, \mathbf{c}^*, \pi^*)$ such that $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}, \mathbf{C}^*, \mathbf{c}^*, \pi^*) = 1$ and $\big(\mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}^*), \mathsf{Extr}(ek, \mathbf{c}^*)\big) \notin \big\{(M_1, \Sigma_1), \ldots, (M_n, \Sigma_n)\big\}$.

This unforgeability notion is reduced to strong unforgeability of **Sig**. On receiving vk, run $(ck, ek) \leftarrow \mathsf{ExSetup}$ and give (ck, ek, vk) to the adversary. Answer a query for $\mathbf{C}$ as follows: using ek, extract M, and query it to the signing oracle to receive Σ; then run $(\mathbf{c}_\Sigma, \pi) \leftarrow \mathsf{SmSigCom}(ck, ek, vk, \mathbf{C}, \Sigma)$ and return $(\mathbf{c}_\Sigma, \pi)$. Since by Definition 5, $\mathsf{SmSigCom}$ and SigCom both commute with $\mathsf{Com}_{\mathcal{M}}$, Com and Prove, this perfectly simulates the adversary's oracle. If the adversary wins the game, we return $\mathsf{Extr}_{\mathcal{M}}(ek, \mathbf{C}^*)$ and $\mathsf{Extr}(ek, \mathbf{c}^*)$ which yields a valid forgery (M, Σ) by perfect soundness of **Proof**.

Note that providing the adversary, who is given the extraction key, with a signing oracle would be redundant: to obtain a signature on a message it suffices to commit to it, submit it to the SigCom oracle and use ek to extract a signature from the reply. The notion is thus stronger than unforgeability of P-signatures [BCKL08], where the adversary is *only* given a signing oracle.

Indistinguishability. The message in $\mathbf{C}$ remains hidden to a signer running SigCom, in a computational sense: replacing ck by $ck^* \leftarrow \mathsf{WISetup}$ is computationally indistinguishable and results in perfectly hiding outputs of Com and $\mathsf{Com}_{\mathcal{M}}$.

Blind Signatures. Given a system of commuting signatures and verifiable encryption, we can easily build a round-optimal blind-signature scheme in the common reference string (CRS) model in a black-box way. The CRS is a commitment key $ck \leftarrow \mathsf{Setup}$. To get a signature on a message M, a user chooses $\mu \leftarrow \mathcal{R}_{\mathcal{M}}$ and sends the commitment $\mathbf{C} := \mathsf{Com}_{\mathcal{M}}(pp, M, \mu)$ to the signer. The latter uses SigCom to produce and send $(\mathbf{c}_\Sigma, \pi)$, a committed signature on M and a proof of validity. The user can produce a proof $\widetilde{\pi} \leftarrow \mathsf{AdPrDC}_{\mathcal{M}}(ck, vk, (M, \mu), \mathbf{c}_\Sigma, \pi)$, which asserts validity of the committed signature on M. The blind signature is defined as $(\mathbf{c}_\Sigma, \widetilde{\pi})$ and is verified by $\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk,M,\cdot)}, \mathbf{c}_\Sigma, \widetilde{\pi})$.

This yields a simpler and generically more efficient construction than that from [Fis06], in which a blind signature consists of a commitment to a signature on $\mathbf{C}$, a commitment to $\mathbf{C}$, proofs of validity of the committed values, and a proof that $\mathbf{C}$ opens to M.

5.3 Additional Properties of Groth-Sahai Proofs

We identify five properties of Groth-Sahai proofs for pairing-product equations (PPE) that will allow us to instantiate commuting signatures. We will refer to the instantiation given in Sections 2.4.2

and 2.4.3. The first property is that proofs are constructed independently of the right-hand side of the equation; if the equation does not contain pairings of two variables, i.e., $\gamma_{ij} = 0$ for all i, j, then they are even independent of the committed values. Given two independent (i.e., with no common variables) equations, and commitments and proofs for them, then the product of the proofs is a proof for the "product of the equations" and the concatenated vectors of commitments. The fourth property states that if we change a committed value by exponentiation then we can adapt the proof. And lastly, given commitments and a proof for an equation, if we commit to a constant of the equation then we can turn the proof into a proof for the set of commitments extended by the new commitment and the equation where the constant is now a variable.

For convenience we restate some equations of Sect. 2.4. A PPE E is defined as

$$\mathrm{E}(X_1, \ldots, X_m; Y_1, \ldots, Y_n) \;:\; \prod_{j=1}^{n} e(A_j, \underline{Y_j}) \prod_{j=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \;=\; \mathfrak{t}_T \;. \tag{5.1}$$

$\mathsf{Prove}\big((\vec{\mathbf{u}}, \vec{\mathbf{v}}), \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n\big)$ is defined by choosing $Z \leftarrow \mathbb{Z}_p^{2\times 2}$ defining for $1 \le k, \ell \le 2$: $t_{k\ell} := \sum_{j=1}^{n} \sum_{i=1}^{m} r_{ik} \gamma_{ij} s_{j\ell}$, and outputting

$$\begin{aligned}
\phi &:= \begin{bmatrix} v_{11}^{t_{11}} v_{21}^{t_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i1}\gamma_{ij}}\big) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ v_{11}^{t_{21}} v_{21}^{t_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big)\big(\prod_{j=1}^n Y_j^{\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \odot (Z \otimes \vec{\mathbf{v}}) \\
\theta &:= \begin{bmatrix} 1 & \big(\prod_{j=1}^n A_j^{s_{j1}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j1}\gamma_{ij}}\big) \\ 1 & \big(\prod_{j=1}^n A_j^{s_{j2}}\big)\big(\prod_{i=1}^m X_i^{\sum_{j=1}^n s_{j2}\gamma_{ij}}\big) \end{bmatrix} \odot (Z \otimes \vec{\mathbf{u}})
\end{aligned} \tag{5.2}$$

5.3.1 Independence of Proofs

In general, proofs are independent of the right-hand side of the equation; moreover, proofs for linear equations are independent of the committed values.

Lemma 3. *For any equation* E *as in* (5.1) *the output of* $\mathsf{Prove}(\cdot, \mathrm{E}, \cdot, \cdot)$ *is independent of* $\mathfrak{t}_T$.

Proof. The result follows by inspection of the proof definition in (5.2), or, more generally, the one in Remark 1 (p. 22), which also encompasses other instantiations of Groth-Sahai proofs. □

For concreteness, we will give the proofs of the next lemmas for the SXDH instantiation, but we note that they also hold for the other instantiations.

Lemma 4. *Proofs for equations for which* $\gamma_{ij} = 0$ *for all* i *and* j *depend only on the randomness of the commitments, but not on the committed values.*

Proof. If for an equation E for all i and j we have $\gamma_{ij} = 0$ then for all k and ℓ also $t_{k,\ell} = 0$. The proof in (5.2) then simplifies to

$$\phi := \begin{bmatrix} 1 & \prod_{i=1}^m B_i^{r_{i1}} \\ 1 & \prod_{i=1}^m B_i^{r_{i2}} \end{bmatrix} \odot (Z \otimes \vec{\mathbf{v}}) \qquad \theta := \begin{bmatrix} 1 & \prod_{j=1}^n A_j^{s_{j1}}) \\ 1 & \prod_{j=1}^n A_j^{s_{j2}}) \end{bmatrix} \odot (Z \otimes \vec{\mathbf{u}})$$

which does not contain values X_i and Y_j. □

5.3.2 Proofs for Composed Equations

Groth-Sahai proofs are homomorphic w.r.t. the equations in the following sense.[4] Given equations

$$\mathrm{E} \;:\; \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \;=\; \mathfrak{t}_T$$
$$\mathrm{E}' \;:\; \prod_{i=1}^{n'} e(A'_i, \underline{Y'_i}) \prod_{i=1}^{m'} e(\underline{X'_i}, B'_i) \prod_{i=1}^{m'}\prod_{j=1}^{n'} e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathfrak{t}'_T$$

and a proof π for commitments $(\vec{\mathbf{c}}, \vec{\mathbf{d}})$ for equation E and a proof π' for commitments $(\vec{\mathbf{c}}', \vec{\mathbf{d}}')$ for equation E′, then $\pi'' := \pi \circ \pi'$ is a proof for commitments $((\vec{\mathbf{c}}, \vec{\mathbf{c}}'), (\vec{\mathbf{d}}, \vec{\mathbf{d}}'))$ and equation E″ defined as follows (for arbitrary $\mathfrak{t}''_T \in \mathbb{G}_T$):

$$\mathrm{E}'' \;:\; \prod_{i=1}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{n'} e(A'_i, \underline{Y'_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m'} e(\underline{X'_i}, B'_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} \prod_{i=1}^{m'}\prod_{j=1}^{n'} e(\underline{X'_i}, \underline{Y'_j})^{\gamma'_{i,j}} \;=\; \mathfrak{t}''_T$$

Lemma 5. *Let* E, E' *and* E'' *be defined as above. If* $\pi = \mathsf{Prove}(ck, \mathrm{E}, (X_i, r_i)_{i=1}^{m}, (Y_j, s_j)_{j=1}^{n};\ Z)$ *and* $\pi' = \mathsf{Prove}(ck, \mathrm{E}', (X'_i, r'_i)_{i=1}^{m'}, (Y'_j, s'_j)_{j=1}^{n'};\ Z')$ *then*

$$\pi \circ \pi' = \mathsf{Prove}(ck, \mathrm{E}'', (X_i, r_i)_{i=1}^{m}, (X'_i, r'_i)_{i=1}^{m'}, (Y_j, s_j)_{j=1}^{n}, (Y'_j, s'_j)_{j=1}^{n'};\ Z + Z') \ .$$

Proof. Equation E″ over $(X_1, \ldots, X_m, X'_1, \ldots, X'_{m'}; Y_1, \ldots, Y_n, Y'_1, \ldots, Y'_{n'})$ is determined by the constants $\vec{A}'' := (\vec{A}, \vec{A}')$, $\vec{B}'' := (\vec{B}, \vec{B}')$ and $\Gamma'' := ((\Gamma, 0), (0, \Gamma'))^{\top}$. The proof $\pi'' = (\phi'', \theta'') := \pi \circ \pi'$ looks as follows (with $t''_{k\ell} := t_{k\ell} + t'_{k\ell}$)

$$\phi'' \;:=\; \begin{bmatrix} v_{11}^{t''_{11}} v_{21}^{t''_{12}} & \big(\prod_{i=1}^{m} B_i^{r_{i1}}\big)\big(\prod_{i=1}^{m'} (B'_i)^{r'_{i1}}\big)\big(\prod_{j=1}^{n} Y_j^{\sum_{i=1}^{m} r_{i1}\gamma_{ij}}\big)\big(\prod_{j=1}^{n'} (Y'_j)^{\sum_{i=1}^{m'} r'_{i1}\gamma'_{ij}}\big) v_{12}^{t''_{11}} v_{22}^{t''_{12}} \\ v_{11}^{t''_{21}} v_{21}^{t''_{22}} & \big(\prod_{i=1}^{m} B_i^{r_{i2}}\big)\big(\prod_{i=1}^{m'} (B'_i)^{r'_{i2}}\big)\big(\prod_{j=1}^{n} Y_j^{\sum_{i=1}^{m} r_{i2}\gamma_{ij}}\big)\big(\prod_{j=1}^{n'} (Y'_j)^{\sum_{i=1}^{m'} r'_{i2}\gamma'_{ij}}\big) v_{12}^{t''_{21}} v_{22}^{t''_{22}} \end{bmatrix} \circ ((Z + Z') \otimes \vec{\mathbf{v}})$$

$$\theta'' \;:=\; \begin{bmatrix} 1 & \big(\prod_{j=1}^{n} A_j^{s_{j1}}\big)\big(\prod_{j=1}^{n'} (A'_j)^{s'_{j1}}\big)\big(\prod_{i=1}^{m} X_i^{\sum_{j=1}^{n} s_{j1}\gamma_{ij}}\big)\big(\prod_{i=1}^{m'} (X'_i)^{\sum_{j=1}^{n'} s'_{j1}\gamma'_{ij}}\big) \\ 1 & \big(\prod_{j=1}^{n} A_j^{s_{j2}}\big)\big(\prod_{j=1}^{n'} (A'_j)^{s'_{j2}}\big)\big(\prod_{i=1}^{m} X_i^{\sum_{j=1}^{n} s_{j2}\gamma_{ij}}\big)\big(\prod_{i=1}^{m'} (X'_i)^{\sum_{j=1}^{n'} s'_{j2}\gamma'_{ij}}\big) \end{bmatrix} \circ ((Z + Z') \otimes \vec{\mathbf{u}})$$

which is a proof for $(\vec{\mathbf{c}}, \vec{\mathbf{c}}', \vec{\mathbf{d}}, \vec{\mathbf{d}}')$ for E″ with internal randomness $Z + Z'$. □

5.3.3 Changing the Committed Value and Adapting Proofs

We give a special case which we require to randomize commitments to *non-trivial* messages in Sect. 5.5.1 (p. 63). We start with some notation. Let $\vec{\mathbf{w}} \in \mathbb{G}^{m \times n}$, $Z \in \mathbb{Z}_p^{m \times n}$ and $k \in \mathbb{Z}_p$. Then by

[4] The homomorphic property of Groth-Sahai proofs for *linear* equations was noted independently in [DHLW10].

$\vec{\mathbf{w}}^k$ we denote componentwise exponentiation, and by $k \cdot Z$ we denote standard scalar multiplication, i.e.,

$$\text{if } \vec{\mathbf{w}} = (w_{ij})_{\substack{1\le i\le m\\ 1\le j\le n}} \text{ then } \vec{\mathbf{w}}^k = (w_{ij}^k)_{\substack{1\le i\le m\\ 1\le j\le n}} \qquad \text{if } Z = (z_{ij})_{\substack{1\le i\le m\\ 1\le j\le n}} \text{ then } k \cdot Z = (k\, z_{ij})_{\substack{1\le i\le m\\ 1\le j\le n}}$$

Consider equation $\mathrm{E}^*(X, Y) : e(\underline{X}, \underline{Y}) = \mathbf{t}_T$; given a proof π for E^*, $\mathsf{Com}(ck, X, r)$ and $\mathsf{Com}(ck, Y, s)$, then π^k is a proof for $e(\underline{X}, \underline{Y}) = \mathbf{t}_T^k$ and $\mathsf{Com}(ck, X^k, k \cdot r)$ and $\mathsf{Com}(ck, Y, s)$.

Lemma 6. *If* $\pi = \mathsf{Prove}\big(ck, \mathrm{E}^*, (X, r), (Y, s);\, Z\big)$ *then* $\pi^k = \mathsf{Prove}\big(ck, \mathrm{E}^*, (X^k, k \cdot r), (Y, s);\, k \cdot Z\big)$.

Proof. By (2.3) on p. 21, we have $(Z \otimes \vec{\mathbf{u}})^k = (k \cdot Z) \otimes \vec{\mathbf{u}}$ and $(Z \otimes \vec{\mathbf{v}})^k = (k \cdot Z) \otimes \vec{\mathbf{v}}$ for $Z \in \mathbb{Z}_p^{2\times 2}$ and $k \in \mathbb{Z}_p$. The proof $\mathsf{Prove}(ck, \mathrm{E}^*, (X, r), (Y, s);\, Z)$ is defined as

$$\pi_1 = \begin{bmatrix} v_{11}^{r_1 s_1} v_{21}^{r_1 s_2} & Y^{r_1} v_{12}^{r_1 s_1} v_{22}^{r_1 s_2} \\ v_{11}^{r_2 s_1} v_{21}^{r_2 s_2} & Y^{r_2} v_{12}^{r_2 s_1} v_{22}^{r_2 s_2} \end{bmatrix} \circ (Z \otimes \vec{\mathbf{v}}) \qquad \pi_2 = \begin{bmatrix} 1 & X^{s_1} \\ 1 & X^{s_2} \end{bmatrix} \circ (Z \otimes \vec{\mathbf{u}})$$

so we have

$$\pi_1^k = \begin{bmatrix} v_{11}^{kr_1 s_1} v_{21}^{kr_1 s_2} & Y^{kr_1} v_{12}^{kr_1 s_1} v_{22}^{kr_1 s_2} \\ v_{11}^{kr_2 s_1} v_{21}^{kr_2 s_2} & Y^{kr_2} v_{12}^{kr_2 s_1} v_{22}^{kr_2 s_2} \end{bmatrix} \circ \big((k \cdot Z) \otimes \vec{\mathbf{v}}\big) \qquad \pi_2^k = \begin{bmatrix} 1 & (X^k)^{s_1} \\ 1 & (X^k)^{s_2} \end{bmatrix} \circ \big((k \cdot Z) \otimes \vec{\mathbf{u}}\big)$$

which is the definition of $\mathsf{Prove}\big(ck, \mathrm{E}^*, (X^k, k \cdot r), (Y, s);\, k \cdot Z\big)$. □

5.3.4 Committing to Constants and Adapting Proofs

Given a proof for an equation, one can commit to one of the constants and adapt the proof. Consider an equation $\mathrm{E}(X_1, \ldots, X_m;\, Y_1, \ldots, Y_n)$ as in (5.1) and a proof (ϕ, θ) for commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_m;\, \mathbf{d}_1, \ldots, \mathbf{d}_n)$. Some calculation shows that (ϕ, θ) is also a proof for equation

$$\mathrm{E}'(\vec{X}, A_k;\, \vec{Y}) \;:\; \prod_{\substack{i=1\\ i\neq k}}^{n} e(A_i, \underline{Y_i}) \prod_{i=1}^{m} e(\underline{X_i}, B_i) \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}}\, e(\underline{A_k}, \underline{Y_k}) \;=\; \mathbf{t}_T$$

and commitments $(\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathsf{Com}(ck, A_k, 0);\, \mathbf{d}_1, \ldots, \mathbf{d}_n)$. This yields the following result.

Lemma 7. *Let* $\pi \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n\big)$ *and for all* i, j *let* $\mathbf{c}_i = \mathsf{Com}(ck, X_i, r_i)$ *and* $\mathbf{d}_j = \mathsf{Com}(ck, Y_j, s_j)$. *Then* $\mathsf{RdProof}\big(ck, \mathrm{E}', (\mathbf{c}_i, 0)_{i=1}^m, (\mathsf{Com}(ck, A_k, 0), r), (\mathbf{d}_j, 0)_{j=1}^n, \pi\big)$ *yields a proof that is distributed as the output of* $\mathsf{Prove}\big(ck, \mathrm{E}', (X_i, r_i)_{i=1}^m, (A_k, r), (Y_j, s_j)_{j=1}^n\big)$. *An analogous result holds for committing to a constant* $B_k \in \mathbb{G}_2$.

5.4 Instantiation of Commuting Signatures

In Sect. 4.2 on p. 39, we constructed a blind signature scheme **BSig** from the scheme **Sig** (Sect. 4.1, p. 37) as follows. The user, who wishes to obtain a signature on a message $(M, N) \in \mathcal{DH} := \{(G^m, H^m) \mid m \in \mathbb{Z}_p\}$, chooses a random $t \leftarrow \mathbb{Z}_p$ and *blinds* the first message component by the factor T^t. The user then sends the following: $U := T^t \cdot M$, commitments $\mathbf{c}_M$ and $\mathbf{c}_N$ to M and N, respectively, and commitments $\mathbf{c}_P$ and $\mathbf{c}_Q$ to G^t and H^t, respectively; moreover, proofs π_M, π_P and

π_U of well-formedness of (M, N), (P, Q) and U, respectively: π_M proves that $e(M, H) = e(G, N)$; π_P proves that $e(P, H) = e(G, Q)$ and π_U proves that $U = T^t \cdot M$, or, equivalently, $e(U, H) = e(T, Q)\, e(M, N)$.

The signer replies with a "pre-signature" on U (which is constructed as a signature on U, but on a message that lacks the second component):

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r$$

We have $A = (K \cdot T^r \cdot U)^{1/(x+c)} = (K \cdot T^{r+t} \cdot M)^{1/(x+c)}$, which is the first component of a signature on (M, N) with randomness $r + t$. Knowing t, the user can fabricate an actual signature on (M, N) from this "pre-signature" by setting $R := R' \cdot G^t = G^{r+t}$ and $S := S' \cdot H^t = H^{r+t}$. Then (A, B, D, R, S) is a signature on (M, N) with randomness $(c, r + t)$. To prevent linking a signature to the signing session, the blind signature is defined as a Groth-Sahai proof of knowledge of the signature. Now to turn this into a commuting signature, there are two key observations.

1. The values $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ which the user sends to the signer can actually be considered as a *commitment* to the message (M, N), which is extractable and randomizable, and which perfectly hides the message when the values are produced using a key $ck^* \leftarrow \mathsf{WISetup}$.

2. Since **Com** is homomorphic, the values $\mathbf{c}_P$ and $\mathbf{c}_Q$ from the newly defined commitment to (M, N) can be used by the signer to produce commitments on the actual signature components R and S. Moreover, we show how π_P and π_U can be used to produce a proof of validity of the committed values using the results from Lemmas 3, 4, 5 and 7 from the last section.

For the blind signature scheme **BSig** in Sect. 4.2, the values $\mathbf{c}_P, \mathbf{c}_Q, \pi_P$ and π_U are mainly used in the proof of unforgeability, when the simulator needs to extract the message, query it to its signing oracle and then use the values P and Q to turn the signature into a pre-signature. We show that all these values can be directly used by the signer to produce commitments to the signature components and even a proof of validity.

5.4.1 Commitments to Messages

We define a *commitment* on a message $(M, N) \in \mathcal{DH}$ as the values $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ the user sends to the signer in the issuing protocol for the blind signature scheme **BSig**. We then show how to randomize a commitment and how to extract the committed value. Since the committed values are the messages of **Sig**, in addition to ck the algorithms also get the signature parameters $pp_{\mathrm{S}} = (F, K, T)$ as input. The randomness space is $\mathcal{R}_{\mathcal{M}} := \mathbb{Z}_p^9$.

$\mathsf{Com}_{\mathcal{M}}$ has inputs $pp = (ck, F, K, T)$, $(M, N) \in \mathcal{DH}$ and $(t, \mu, \nu, \rho, \sigma) \in \mathbb{Z}_p \times \mathcal{R}^4 \cong \mathbb{Z}_p^9$. We define the following equations:

$$\mathrm{E}_{\mathcal{DH}}(M, N) \;:\; e(G^{-1}, \underline{N})\, e(\underline{M}, H) = 1_T \tag{5.3}$$

$$\mathrm{E}_U(M, Q) \;:\; e(T^{-1}, \underline{Q})\, e(\underline{M}, H^{-1}) = e(U, H)^{-1} \tag{5.4}$$

$\mathsf{Com}_{\mathcal{M}}(pp, (M, N), (t, \mu, \nu, \rho, \sigma))$ defines $P = G^t$ and $Q = H^t$, computes

$$\begin{array}{lll} \mathbf{c}_M := \mathsf{Com}(ck, M, \mu) & \mathbf{c}_N := \mathsf{Com}(ck, N, \nu) & \pi_M \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}},\ (M, \mu), (N, \nu)\big) \\ \mathbf{c}_P := \mathsf{Com}(ck, P, \rho) & \mathbf{c}_Q := \mathsf{Com}(ck, Q, \sigma) & \pi_P \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}},\ (P, \rho), (Q, \sigma)\big) \\ U := T^t \cdot M & & \pi_U \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_U,\ (M, \mu), (Q, \sigma)\big) \end{array}$$

and returns $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$.

$\mathcal{C}_{\mathcal{M}}(pp)$, the space of valid $\mathbf{Com}_{\mathcal{M}}$ commitments under parameters $pp = (ck, pp_{\mathrm{S}})$ is defined as follows (with $\mathcal{C}_1 := \mathbb{G}_1^2$, $\mathcal{C}_2 := \mathbb{G}_2^2$, $\mathcal{P} := \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$):

$$\mathcal{C}_{\mathcal{M}}(pp) \;:=\; \big\{(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U) \in (\mathcal{C}_1 \times \mathcal{C}_2 \times \mathcal{P})^2 \times \mathbb{G}_1 \times \mathcal{P} \cong \mathbb{G}_1^{17} \times \mathbb{G}_2^{16} \;\mid\; \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_M, \mathbf{c}_N, \pi_M) \wedge \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_P, \mathbf{c}_Q, \pi_P) \wedge \mathsf{Verify}(ck, \mathrm{E}_U, \mathbf{c}_M, \mathbf{c}_Q, \pi_U)\big\} \;.$$

$\mathsf{RdCom}_{\mathcal{M}}$, on input (ck, pp_{S}), $\mathbf{C}$ and randomness $(t', \mu', \nu', \rho', \sigma') \in \mathcal{R}_{\mathcal{M}}$, first defines

$$\hat{\mathbf{c}}_P := \mathbf{c}_P \circ \mathsf{Com}(ck, G^{t'}, 0) \qquad \hat{\mathbf{c}}_Q := \mathbf{c}_Q \circ \mathsf{Com}(ck, H^{t'}, 0) \qquad U' := U \cdot T^{t'}$$

This replaces randomness t by $t + t'$. Then it sets

$$\begin{aligned} \mathbf{c}'_M &:= \mathsf{RdCom}(ck, \mathbf{c}_M, \mu') & \pi'_M &\leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\mathbf{c}_M, \mu'), (\mathbf{c}_N, \nu'), \pi_M) \\ \mathbf{c}'_N &:= \mathsf{RdCom}(ck, \mathbf{c}_N, \nu') & \pi'_P &\leftarrow \mathsf{RdProof}(ck, \mathrm{E}_{\mathcal{DH}}, (\hat{\mathbf{c}}_P, \rho'), (\hat{\mathbf{c}}_Q, \sigma'), \pi_P) \\ \mathbf{c}'_P &:= \mathsf{RdCom}(ck, \hat{\mathbf{c}}_P, \rho') & \pi'_U &\leftarrow \mathsf{RdProof}(ck, \mathrm{E}_U, (\mathbf{c}_M, \mu'), (\hat{\mathbf{c}}_Q, \sigma'), \pi_U) \\ \mathbf{c}'_Q &:= \mathsf{RdCom}(ck, \hat{\mathbf{c}}_Q, \sigma') \end{aligned}$$

which replaces randomness (μ, ν, ρ, σ) by $(\mu + \mu', \nu + \nu', \rho + \rho', \sigma + \sigma')$. Finally, it returns $\mathbf{C}' = (\mathbf{c}'_M, \mathbf{c}'_N, \pi'_M, \mathbf{c}'_P, \mathbf{c}'_Q, \pi'_P, U', \pi'_U) \in \mathcal{C}_{\mathcal{M}}(pp)$.

$\mathsf{Extr}_{\mathcal{M}}$, on input ek and $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ returns $\big(\mathsf{Extr}(ek, \mathbf{c}_M), \mathsf{Extr}(ek, \mathbf{c}_N)\big)$.

Theorem 7. $\mathbf{Com}_{\mathcal{M}}$ *is a randomizable extractable commitment scheme that is perfectly binding and computationally hiding.*

Proof. The commitment $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ is binding by the binding property of SXDH commitments. A correctly constructed commitment contains valid proofs; in particular, we have $e(U, H) = e(T^t, H)\, e(M, H) = e(T, Q)\, e(M, H)$, thus (5.4) is satisfied.

The scheme is computationally hiding as defined Sect. 2.2.1 (p. 14). Let $ck^* \leftarrow \mathsf{WISetup}$. Then for every $(M, N) \in \mathcal{DH}$ there exists t s.t. $U = T^t \cdot M$. Moreover there exist μ, ν, ρ and σ s.t. $\mathbf{c}_M = \mathsf{Com}(ck, M, \mu)$, $\mathbf{c}_N := \mathsf{Com}(ck, N, \nu)$, $\mathbf{c}_P := \mathsf{Com}(ck, G^t, \rho)$, and $\mathbf{c}_Q := \mathsf{Com}(ck, H^t, \sigma)$. So for every $\mathbf{C}$ and every $(M, N) \in \mathcal{DH}$ there exists $\kappa := (t, \mu, \nu, \rho, \sigma) \in \mathcal{R}_{\mathcal{M}}$ s.t. $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck^*, (M, N), \kappa)$.

Moreover, $\mathsf{RdCom}_{\mathcal{M}}$ randomizes a commitment. When $(U, \mathbf{c}_P, \mathbf{c}_Q)$ is replaced by $(U', \hat{\mathbf{c}}_P, \hat{\mathbf{c}}_Q)$ in the first step, t is replaced by $t + t'$ (since the commitments are homomorphic, $\hat{\mathbf{c}}_P$ is a commitment to $P \cdot G^{t'}$ and $\mathbf{c}'_Q$ commits to $Q \cdot H^{t'}$; note that, by Lemma 4, π_P and π_U do not depend on t but only on the randomness of the commitments—which is not changed in the first step.) In the second step, (μ, ν, ρ, σ) is replaced by $(\mu + \mu', \nu + \nu', \rho + \rho', \sigma + \sigma')$. □

5.4.2 Making Committents to a Signature on a Committed Message and a Proof of Validity

We show how the signer can use the values in $\mathbf{C}$ to produce a proof of knowledge

$$(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R) \;\in\; \mathcal{C}_1 \times \mathcal{C}_1 \times \mathcal{C}_2 \times \mathcal{C}_1 \times \mathcal{C}_2 \times (\mathcal{P})^3 \;\cong\; \mathbb{G}_1^{18} \times \mathbb{G}_2^{16}$$

of a signature (A, B, D, R, S) (i.e., a verifiably encrypted signature) on the message committed in $\mathbf{C}$. The proofs π_A, π_B and π_R attest that the values committed in $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ and $\mathbf{c}_M$ from $\mathbf{C}$ satisfy the verification equations in (4.2) (p. 37), respectively, i.e.,

$$\begin{aligned} \mathrm{E}_A(A, M;\, S, D) &: e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\ \mathrm{E}_B(B;\, D) &: e(F^{-1}, \underline{D})\, e(\underline{B}, H) = 1_T \\ \mathrm{E}_R(R;\, S) &: e(G^{-1}, \underline{S})\, e(\underline{R}, H) = 1_T \end{aligned} \tag{5.5}$$

In the blind signature **BSig** from Sect. 4.2, on receiving $\mathbf{C}$, the signer checks the proofs contained in it, and then produces a pre-signature by choosing $c, r \leftarrow \mathbb{Z}_p$ and computing

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad B := F^c \qquad D := H^c \qquad R' := G^r \qquad S' := H^r$$

The user knowing t s.t. $U = T^t \cdot M$ can turn these values into a signature by setting $R := R' \cdot G^t$ and $S := S' \cdot H^t$. Since the commitments are homomorphic, the signer can—without knowledge of the values $P = G^t$ and $Q = H^t$, but knowing the commitments $\mathbf{c}_P$ and $\mathbf{c}_Q$—make *commitments* to R and S:

$$\mathbf{c}_R := \mathbf{c}_P \circ \mathsf{Com}(ck, R', 0) = \mathsf{Com}(ck, R, \rho) \qquad \mathbf{c}_S := \mathbf{c}_Q \circ \mathsf{Com}(ck, S', 0) = \mathsf{Com}(ck, S, \sigma) \tag{5.6}$$

The signer then chooses $\alpha, \beta, \delta \leftarrow \mathcal{R}$, and makes the remaining commitments:

$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_B := \mathsf{Com}(ck, B, \beta) \qquad \mathbf{c}_D := \mathsf{Com}(ck, D, \delta) \tag{5.7}$$

The vector $\vec{\mathbf{c}}_\Sigma := (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ is thus a commitment to the *actual* signature (A, B, D, R, S). It remains to construct proofs π_A, π_B and π_R that the committed values satisfy the 3 equations in (5.5)—without knowledge of μ, ρ and σ, the randomness of the commitments $\mathbf{c}_M, \mathbf{c}_R$ and $\mathbf{c}_S$, respectively! This can be done using the following observations:

1. Equation $\mathrm{E}_R(R;\, S)$ is actually $\mathrm{E}_{\mathcal{DH}}(R;\, S)$ from (5.3). Since by (5.6) $\mathbf{c}_R$ and $\mathbf{c}_P$ have the same randomness ρ, and $\mathbf{c}_S$ and $\mathbf{c}_Q$ have the same randomness σ, and since by Lemma 4, a proof for the equation $\mathrm{E}_{\mathcal{DH}}$ is independent of the committed values, we can set $\pi_R := \pi_P$.

2. Lemmas 3 and 4 yield that proofs for Equation E_U only depend on the randomness of the commitments. Since $\mathbf{c}_S = \mathsf{Com}(ck, S, \sigma)$ and $\mathbf{c}_Q = \mathsf{Com}(ck, Q, \sigma)$ have the same randomness, π_U is not only a proof for $\mathrm{E}_U(M; Q)$ but also for $\mathrm{E}_U(M; S)$ for $\mathbf{c}_M$ and $\mathbf{c}_S$. Moreover, define

   $$\mathrm{E}_{A^\dagger}(A;\, D) \;:\; e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = 1_T \tag{5.8}$$

   and let $\pi_{A^\dagger} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A, \alpha), (D, \delta))$, which can be computed by the signer who chose α and δ. Since the product of the left-hand sides of $\mathrm{E}_U(M; S)$ (Equation (5.4)) and $\mathrm{E}_{A^\dagger}(A; D)$ is the left-hand side of $\mathrm{E}_A(A, M;\, S, D)$, by Lemma 5 we have $\pi_A := \pi_U \circ \pi_{A^\dagger}$ is a proof for E_A.

The remaining proof π_B can be constructed regularly, since randomness (β, δ) is known to the signer. Finally, to get a *random* proof of knowledge, the signer randomizes all commitments and proofs using RdCom and $\mathsf{RdProof}$ as defined in Sect. 2.4.3. Algorithm SigCom is summarized in Figure 5.1. We now formally prove that the output of SigCom is distributed as required by Definition 5.

Theorem 8. SigCom*, as defined in Figure 5.1, is commuting.*

SigCom$(ck, sk, \mathbf{C})$. Parse $\mathbf{C}$ as $(\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and sk as x. If π_M, π_P and π_U are valid then choose $c, r \leftarrow \mathbb{Z}_p$ and $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathbb{Z}_p^2$ and compute the following values:

$$A := (K \cdot T^r \cdot U)^{\frac{1}{x+c}} \qquad \mathbf{c}_B := \mathsf{Com}(ck, F^c, \beta) \qquad \mathbf{c}_R := \mathbf{c}_P \circ \mathsf{Com}(ck, G^r, \rho')$$
$$\mathbf{c}_A := \mathsf{Com}(ck, A, \alpha) \qquad \mathbf{c}_D := \mathsf{Com}(ck, H^c, \delta) \qquad \mathbf{c}_S := \mathbf{c}_Q \circ \mathsf{Com}(ck, H^r, \sigma')$$

$\pi'_A := \pi_U \circ \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger}, (A, \alpha), (H^c, \delta); 0)$ (with $\mathrm{E}_{A^\dagger}$ being Equation (5.8))

$\pi_A \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_A, (\mathbf{c}_A, 0), (\mathbf{c}_D, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, \sigma'), \pi'_A)$

$\pi_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R, \rho'), (\mathbf{c}_S, \sigma'), \pi_P)$ $\qquad \pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (F^c, \beta), (H^c, \delta))$

Return $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$.

Figure 5.1: Making commitments to a signature and proving knowledge.

Proof. The algorithm SigCom optimizes the steps discussed informally above by constructing and randomizing $\mathbf{c}_R$ and $\mathbf{c}_S$ in one step. Moreover, $\mathbf{c}_A, \mathbf{c}_B$ and $\mathbf{c}_D$ need not be randomized since SigCom chooses their randomness; in addition, being produced "freshly", π_B need not be randomized either. We formally prove that the output of the algorithm is correctly distributed. Recall that $\mathsf{Prove}(\ldots; Z)$ denotes the output of Prove when $Z \in \mathbb{Z}_p^{2\times 2}$ is the employed internal randomness.

Let $\mathbf{C} = (\mathbf{c}_M, \mathbf{c}_N, \pi_M, \mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_U)$ and let $(M, N) \in \mathcal{DH}$ and $rand := (t, \mu, \nu, \rho, \sigma) \in \mathcal{R}_{\mathcal{M}}$ be such that $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck, (M, N), rand)$; in particular, since $\mathbf{Com}_{\mathcal{M}}$ is perfectly binding, we have $U = T^t \cdot M$ and

$$\mathbf{c}_M = \mathsf{Com}(ck, M, \mu) \qquad \mathbf{c}_N = \mathsf{Com}(ck, N, \nu) \qquad \mathbf{c}_P = \mathsf{Com}(ck, G^t, \rho) \qquad \mathbf{c}_Q = \mathsf{Com}(ck, H^t, \sigma)$$

Let moreover Z_M, Z_P and Z_U be such that

$$\pi_M := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (M, \mu), (N, \nu);\ Z_M\big)$$
$$\pi_P := \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (G^t, \rho), (H^t, \sigma);\ Z_P\big) \qquad \pi_U := \mathsf{Prove}\big(ck, \mathrm{E}_U, (M, \mu), (H^t, \sigma);\ Z_U\big)$$

Correctness. Let $c, r, \alpha, \beta, \delta, \rho', \sigma'$ be the values chosen by SigCom. We have

$$\mathbf{c}_A = \mathsf{Com}(ck, (K \cdot T^{r+t} \cdot M)^{\frac{1}{x+c}}, \alpha) \qquad \mathbf{c}_B = \mathsf{Com}(ck, F^c, \beta) \qquad \mathbf{c}_D = \mathsf{Com}(ck, H^c, \delta)$$
$$\mathbf{c}_R = \mathsf{Com}(ck, G^{t+r}, \rho + \rho') \qquad \mathbf{c}_S = \mathsf{Com}(ck, H^{t+r}, \sigma + \sigma')$$

where the first equation follows from the definition of U and the last two from the homomorphic property of Com. Let (A, B, D, R, S) be the values committed in $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$, which compose a valid signature; in particular

$$(A, B, D, R, S) = \mathsf{Sign}(x, (M, N);\ (c, r + t)) \ .$$

Define π'_A as in Figure 5.1 and let $\pi'_R := \pi_P$. In the discussion above, using Lemma 5, we showed the following:

$$\pi'_A = \mathsf{Prove}\big(ck, \mathrm{E}_A, (A, \alpha), (M, \mu), (S, \sigma), (D, \delta);\ Z_U\big) \quad \pi'_R = \mathsf{Prove}\big(ck, \mathrm{E}_{\mathcal{DH}}, (P, \rho), (Q, \sigma);\ Z_P\big)$$

Let Z_A, Z_B and Z_R be the randomness used by RdProof and Prove in the construction of π_A, π_B and π_R, respectively. By the properties of RdProof (cf. Remark 1 on p. 22) we have the following:

$$\pi_A := \mathsf{Prove}(ck, \mathrm{E}_A, (A, \alpha), (M, \mu), (S, \sigma + \sigma'), (D, \delta);\ Z_U + Z_A + Z')$$
$$\pi_R := \mathsf{Prove}(ck, \mathrm{E}_R, (R, \rho + \rho'), (S, \sigma + \sigma');\ Z_P + Z_R + Z'')$$
$$\pi_B := \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (B, \beta), (D, \delta);\ Z_B)$$

where the Z' is defined by α, μ and σ' and Z'' is defined by ρ and σ (cf. equation (2.6) in Remark 1).

The resulting output $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S, \pi_A, \pi_B, \pi_R)$ of SigCom is thus the same as the values constructed the following way:

$$\begin{aligned}
(A, B, D, R, S) &:= \Sigma = \mathsf{Sign}(x, (M, N), (\hat{c}, \hat{r})) \\
(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S) &:= \mathsf{Com}(ck, (A, B, D, R, S), (\hat{\alpha}, \hat{\beta}, \hat{\delta}, \hat{\rho}, \hat{\sigma})) \\
\pi_A &:= \mathsf{Prove}\big(ck, \mathrm{E}_A, (A, \hat{\alpha}), (M, \mu), (S, \hat{\sigma}), (D, \hat{\delta});\ \hat{Z}_A\big) \\
\pi_B &:= \mathsf{Prove}\big(ck, \mathrm{E}_B, (B, \hat{\beta}), (D, \hat{\delta});\ \hat{Z}_B\big) \\
\pi_R &:= \mathsf{Prove}\big(ck, \mathrm{E}_R, (R, \hat{\rho}), (S, \hat{\sigma});\ \hat{Z}_R\big)
\end{aligned}$$

when $(\hat{\alpha}, \hat{\beta}, \hat{\delta}, \hat{\rho}, \hat{\sigma}, \hat{Z}_A, \hat{Z}_B, \hat{Z}_R)$ are defined as

$$\begin{array}{lllll}
\hat{c} = c & \hat{r} = r + t & \hat{\alpha} = \alpha & \hat{\beta} = \beta & \hat{\delta} = \delta \\
\hat{\rho} = \rho + \rho' & \hat{\sigma} = \sigma + \sigma' & \hat{Z}_A = Z_U + Z_A + Z' & \hat{Z}_B = Z_B & \hat{Z}_R = Z_P + Z_R + Z''
\end{array}$$

All these values are uniformly random since $c, r, \alpha, \beta, \delta, \rho', \sigma', Z_A, Z_B$, and Z_R are chosen uniformly and independently at random by SigCom. □

Instantiation of SmSigCom. This algorithm is similar to SigCom but instead of the signing key sk it is directly given a signature (A, B, D, R, S). It proceeds like SigCom but starting from a signature instead of producing a pre-signature: choose $\alpha, \beta, \delta, \rho', \sigma' \leftarrow \mathcal{R}$ and set $\mathbf{c}_A, \mathbf{c}_B$ and $\mathbf{c}_D$ as in (5.7); use ek to extract P and Q from $\mathbf{C}$ and set

$$\begin{aligned}
\mathbf{c}_R &:= \mathbf{c}_P \circ \mathsf{Com}(ck, R \cdot P^{-1}, \rho') = \mathsf{Com}(ck, R, \rho + \rho') \\
\mathbf{c}_S &:= \mathbf{c}_Q \circ \mathsf{Com}(ck, S \cdot Q^{-1}, \sigma') = \mathsf{Com}(ck, S, \sigma + \sigma')
\end{aligned}$$

Now π_A, π_B and π_R can be computed as in SigCom in Figure 5.1.

5.4.3 Instantiations of Proof Adaptation for Committing and Decommitting

We define equations $\mathrm{E}_{\widetilde{A}}$ and $\mathrm{E}_{\bar{A}}$ and recall E_A, which all represent the first verification equation in (4.2) but with different group elements being variables.

$$\begin{aligned}
\mathrm{E}_A(A, M;\ S, D) &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K, H) \\
\mathrm{E}_{\widetilde{A}}(A;\ S, D) &:\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = e(K \cdot M, H) \\
\mathrm{E}_{\bar{A}}(M) &:\ e(\underline{M}, H^{-1}) = e(A, Y \cdot D)^{-1} e(K, H)\, e(T, S)
\end{aligned}$$

Recall equations E_B and E_R from (5.5). Then we have

$$\begin{aligned}
\mathrm{E}_{\mathsf{Ver}((X,Y),\cdot,\cdot)}((M, N), (A, B, D, R, S)) &\equiv \mathrm{E}_A(A, M;\ S, D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \\
\mathrm{E}_{\mathsf{Ver}((X,Y),(M,N),\cdot)}(A, B, D, R, S) &\equiv \mathrm{E}_{\widetilde{A}}(A;\ S, D) \wedge \mathrm{E}_B(B;\ D) \wedge \mathrm{E}_R(R;\ S) \\
\mathrm{E}_{\mathsf{Ver}((X,Y),\cdot,(A,B,D,R,S))}(M, N) &\equiv \mathrm{E}_{\bar{A}}(M)
\end{aligned}$$

Since the product of the left-hand sides of $\mathrm{E}_{\widetilde{A}}$ and $\mathrm{E}_{\bar{A}}$ is the left-hand side of E_A, by Lemma 5 we have

$$\pi_A = \pi_{\widetilde{A}} \circ \pi_{\bar{A}}\ , \tag{5.9}$$

which allows us to implement the algorithms AdPrC, $\mathsf{AdPrC}_{\mathcal{M}}$, AdPrDC and $\mathsf{AdPrDC}_{\mathcal{M}}$, defined in Sect. 5.2. AdPrC transforms a proof $\bar{\pi}$ for $\mathbf{C}$ and Σ into a proof π for $\mathbf{C}$ and a commitment to Σ and AdPrDC does the converse. $\mathsf{AdPrC}_{\mathcal{M}}$ transforms a proof $\widetilde{\pi}$ for M and $\mathbf{c}_\Sigma$ into a proof π for a commitment to M and $\mathbf{c}_\Sigma$ whereas $\mathsf{AdPrDC}_{\mathcal{M}}$ does the converse.

Equation (5.9) lets us transform $\pi_{\bar{A}}$ (and $\pi_{\widetilde{A}}$) into π_A and thus proofs $\bar{\pi}$ for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\Sigma)}$ (and proofs $\widetilde{\pi}$ for $\mathrm{E}_{\mathsf{Ver}(vk,(M,N),\cdot)}$) into proofs π for $\mathrm{E}_{\mathsf{Ver}(vk,\cdot,\cdot)}$. Likewise, it lets us transform proofs π into proofs $\bar{\pi}$ or $\widetilde{\pi}$. Note that when a proof for an equation is multiplied with a freshly generated proof, it is uniformly distributed: by Lemma 5, if Z is the internal randomness of the given proof and Z' that of the freshly generated proof then the randomness of the product of proofs is $Z + Z'$. However, when an algorithm reuses a proof from the input it must be randomized first.

$\mathsf{AdPrC}\big(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \bar{\pi}\big)$.

The proof $\bar{\pi}$ is a proof for equation $\mathrm{E}_{\bar{A}}$. The algorithm sets

$$\pi_{\widetilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\widetilde{A}}, (A,\alpha), (S,\sigma), (D,\delta)) \qquad \begin{array}{l}\pi_B \leftarrow \mathsf{Prove}(ck, \mathrm{E}_B, (B,\beta), (D,\delta))\\ \pi_R \leftarrow \mathsf{Prove}(ck, \mathrm{E}_R, (R,\rho), (S,\sigma))\end{array}$$

for E_B and E_R as defined in (5.5). It then returns $\pi := (\pi_{\widetilde{A}} \circ \pi_{\bar{A}}, \pi_B, \pi_R)$.

$\mathsf{AdPrC}_{\mathcal{M}}\big(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), (\mathbf{c}_A,\mathbf{c}_B,\mathbf{c}_D,\mathbf{c}_R,\mathbf{c}_S), \widetilde{\pi}\big)$.

The proof $\widetilde{\pi}$ is of the form $(\pi_{\widetilde{A}}, \pi_B, \pi_R)$. The algorithm sets

$$\pi_{\bar{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\bar{A}}, (M,\mu)) \qquad \begin{array}{l}\pi'_B \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_B, (\mathbf{c}_B,0), (\mathbf{c}_D,0), \pi_B)\\ \pi'_R \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_R, (\mathbf{c}_R,0), (\mathbf{c}_S,0), \pi_R)\end{array} \tag{5.10}$$

and returns $\pi := (\pi_{\widetilde{A}} \circ \pi_{\bar{A}}, \pi'_B, \pi'_R)$.

$\mathsf{AdPrDC}\big(pp, vk, \mathbf{C}, ((A,B,D,R,S),(\alpha,\beta,\delta,\rho,\sigma)), \pi\big)$.

The proof π is of the form (π_A, π_B, π_R). The algorithm sets

$$\pi_{\widetilde{A}} \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\widetilde{A}}, (A,\alpha), (S,\sigma), (D,\delta))$$

and returns $\bar{\pi} := \pi_A \oslash \pi_{\widetilde{A}}$ (where "$\oslash$" denotes componentwise division, that is: replace all the components of the second argument by their inverses and then multiply them with those of the first argument).

$\mathsf{AdPrDC}_{\mathcal{M}}\big(pp, vk, ((M,N),(t,\mu,\nu,\rho,\sigma)), (\mathbf{c}_A,\mathbf{c}_B,\mathbf{c}_D,\mathbf{c}_R,\mathbf{c}_S), \pi\big)$.

The proof π is of the form (π_A, π_B, π_R). The algorithm computes $\pi_{\bar{A}}, \pi'_B$ and π'_R as in (5.10) and returns $\widetilde{\pi} := (\pi_A \oslash \pi_{\bar{A}}, \pi_B, \pi_R)$.

Instantiation of $\mathsf{AdPrC}_{\mathcal{K}}$ and $\mathsf{AdPrDC}_{\mathcal{K}}$. In applications (such as the credentials in Chapter 7) where the signer is to remain anonymous, she makes a commitment

$$\mathbf{c}_{vk} := \big(\mathbf{c}_X = \mathsf{Com}(ck, X, \xi),\ \mathbf{c}_Y = \mathsf{Com}(ck, Y, \psi),\ \pi_X = \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (X,\xi), (Y,\psi))\big)$$

to her public key $vk = (X, Y) \in \mathcal{DH}$ and proves that the values in $\mathbf{c}_\Sigma$ are a valid signature on the value (M, N) in $\mathbf{C}$ under the public key that is committed in $\mathbf{c}_{vk}$. The verification equations representing $\mathrm{E}_{\mathsf{Ver}(\cdot,\cdot,\cdot)}((X,Y),(M,N),(A,B,D,R,S))$, defined in (4.2) (p. 37) are the following:

$$\begin{aligned}\mathrm{E}_{\widehat{A}}(A, M;\ S, Y, D) &: e(T^{-1}, \underline{S})\, e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{Y}) e(\underline{A}, \underline{D}) = e(K, H)\\ \mathrm{E}_B(B;\ D) &: e(F^{-1}, \underline{D})\, e(\underline{B}, H) = 1_T\\ \mathrm{E}_R(R;\ S) &: e(G^{-1}, \underline{S})\, e(\underline{R}, H) = 1_T\end{aligned}$$

Given a commitment $\mathbf{C}$ to a message, a commitment $\mathbf{c}_\Sigma = (\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ to a signature, and a proof $\pi = (\pi_A, \pi_B, \pi_R)$ of validity, π_A can be adapted to $\pi_{\widehat{A}}$ using Lemma 7 from Sect. 5.3 setting

$$\pi_{\widehat{A}} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), \psi), (\mathbf{c}_D, 0), \pi_A\big) \ . \tag{5.11}$$

To adapt a proof to a decommitment of $\mathbf{c}_{vk}$, we have to reset the randomness of $\mathbf{c}_Y$ to 0. $\mathsf{AdPrDC}_{\mathcal{K}}$ does thus the converse: it sets

$$\pi_A \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\widehat{A}}, (\mathbf{c}_A, 0), (\mathbf{c}_M, 0), (\mathbf{c}_S, 0), (\mathsf{Com}(ck, Y, 0), -\psi), (\mathbf{c}_D, 0), \pi_{\widehat{A}}\big) \ .$$

To get a *random* proof, both $\mathsf{AdPrC}_{\mathcal{K}}$ (and $\mathsf{AdPrDC}_{\mathcal{K}}$) also randomize the adapted proof $(\pi_{\widehat{A}}, \pi_B, \pi_R)$ (and (π_A, π_B, π_R)) before outputting it.

Remark 7. For concreteness, we show how a proof π_A for equation E_A is transformed into a proof $\pi_{\widehat{A}}$ for equation $\mathrm{E}_{\widehat{A}}$ for the SXDH instantiation of Groth-Sahai proofs in detail. The definitions of $\pi_{\widehat{A}}$ and $\mathrm{E}_{\widehat{A}}$ are given in Figure 5.2 on the next page. $\mathsf{RdProof}$ in (5.11) chooses $Z' \leftarrow \mathbb{Z}_p^{2\times 2}$ and sets

$$\pi_{\widehat{A},1} \ := \ \pi_{A,1} \circledcirc (Z' \otimes \vec{\mathbf{v}}) \qquad \pi_{\widehat{A},2} \ := \ \pi_{A,2} \circledcirc \begin{bmatrix} \mathbf{c}_{A,1}^{\psi_1} & \mathbf{c}_{A,2}^{\psi_1} \\ \mathbf{c}_{A,1}^{\psi_2} & \mathbf{c}_{A,2}^{\psi_2} \end{bmatrix} \circledcirc (Z' \otimes \vec{\mathbf{u}})$$

Let $Z \in \mathbb{Z}_p^{2\times 2}$ denote the randomness in π_A. We define

$$\hat{Z} \ := \ \begin{bmatrix} z_{11} + z'_{11} + \alpha_1\psi_1 & z_{12} + z'_{12} + \alpha_2\psi_1 \\ z_{21} + z'_{21} + \alpha_1\psi_2 & z_{22} + z'_{22} + \alpha_2\psi_2 \end{bmatrix} \ .$$

Then, substituting π_A by its definition from Figure 5.2, we have

$$\begin{aligned}
\pi_{\widehat{A},1} \ &= \ \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \circledcirc ((Z + Z') \otimes \vec{\mathbf{v}}) \\
&= \ \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \circledcirc \begin{bmatrix} v_{11}^{\alpha_1\psi_1} v_{21}^{\alpha_1\psi_2} & v_{12}^{\alpha_1\psi_1} v_{22}^{\alpha_1\psi_2} \\ v_{11}^{\alpha_2\psi_1} v_{21}^{\alpha_2\psi_2} & v_{12}^{\alpha_2\psi_1} v_{22}^{\alpha_2\psi_2} \end{bmatrix} \circledcirc (\hat{Z} \otimes \vec{\mathbf{v}}) \\
\pi_{\widehat{A},2} \ &= \ \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \circledcirc \begin{bmatrix} (u_{11}^{\alpha_1} u_{21}^{\alpha_2})^{\psi_1} & (A u_{12}^{\alpha_1} u_{22}^{\alpha_2})^{\psi_1} \\ (u_{11}^{\alpha_1} u_{21}^{\alpha_2})^{\psi_2} & (A u_{12}^{\alpha_1} u_{22}^{\alpha_2})^{\psi_2} \end{bmatrix} \circledcirc ((Z + Z') \otimes \vec{\mathbf{u}}) \\
&= \ \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1 + \psi_1} \\ 1 & T^{-\sigma_2} A^{\delta_2 + \psi_2} \end{bmatrix} \circledcirc (\hat{Z} \otimes \vec{\mathbf{u}}) \ ,
\end{aligned}$$

which is a proof for equation $\mathrm{E}_{\widehat{A}}$ when $Z_{\widehat{A}} := \hat{Z}$ (which is uniformly random since Z' was chosen uniformly random).

We summarize the results of this section in the following theorem.

Theorem 9. *Under the ADH-SDH and the SXDH assumption,* $\big(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}, \mathbf{Com}_{\mathcal{M}}, \mathsf{AdPrC}, \mathsf{AdPrDC}, \mathsf{AdPrC}_{\mathcal{M}}, \mathsf{AdPrDC}_{\mathcal{M}}, \mathsf{AdPrC}_{\mathcal{K}}, \mathsf{AdPrDC}_{\mathcal{K}}, \mathsf{SigCom}, \mathsf{SmSigCom}\big)$ *is a system of commuting signatures and verifiable encryption as defined in Definition 5.*

For concreteness, we give the form of the various proofs discussed in this section in the SXDH instantiation in Figure 5.2.

C Proofs contained in a $\mathbf{Com}_{\mathcal{M}}$ commitment, for equations

$$\mathrm{E}_P = \mathrm{E}_M = \mathrm{E}_{\mathcal{DH}}(M,N) : e(G^{-1},\underline{N})\, e(\underline{M},H) = 1_T$$
$$\mathrm{E}_U(M,Q) \;:\; e(T^{-1},\underline{Q})e(\underline{M},H^{-1}) = e(U,H)^{-1}$$

$$\pi_{M,1} = \begin{bmatrix} 1 & H^{\mu_1} \\ 1 & H^{\mu_2} \end{bmatrix} \circledcirc (Z_M \otimes \vec{\mathbf{v}}) \quad \pi_{P,1} = \begin{bmatrix} 1 & H^{\rho_1} \\ 1 & H^{\rho_2} \end{bmatrix} \circledcirc (Z_P \otimes \vec{\mathbf{v}}) \quad \pi_{U,1} = \begin{bmatrix} 1 & H^{-\mu_1} \\ 1 & H^{-\mu_2} \end{bmatrix} \circledcirc (Z_U \otimes \vec{\mathbf{v}})$$

$$\pi_{M,2} = \begin{bmatrix} 1 & G^{-\nu_1} \\ 1 & G^{-\nu_2} \end{bmatrix} \circledcirc (Z_M \otimes \vec{\mathbf{u}}) \quad \pi_{P,2} = \begin{bmatrix} 1 & G^{-\sigma_1} \\ 1 & G^{-\sigma_2} \end{bmatrix} \circledcirc (Z_P \otimes \vec{\mathbf{u}}) \quad \pi_{U,2} = \begin{bmatrix} 1 & T^{-\sigma_1} \\ 1 & T^{-\sigma_2} \end{bmatrix} \circledcirc (Z_U \otimes \vec{\mathbf{u}})$$

$\boxed{\pi_A}$ Proof for equation $\mathrm{E}_A(A,M;\; S,D) \;:\; e(T^{-1},\underline{S})\, e(\underline{A},Y)e(\underline{M},H^{-1})\, e(\underline{A},\underline{D}) = e(K,H)$

$$\pi_{A,1} = \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \circledcirc (Z_A \otimes \vec{\mathbf{v}}) \quad \pi_{A,2} = \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \circledcirc (Z_A \otimes \vec{\mathbf{u}})$$

$\boxed{\pi_{\widetilde{A}}}$ Proof for equation $\mathrm{E}_{\widetilde{A}}(A;\; S,D) \;:\; e(T^{-1},\underline{S})\, e(\underline{A},Y)\, e(\underline{A},\underline{D}) = e(K\cdot M,H)$

$$\pi_{\widetilde{A},1} = \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \circledcirc (Z_{\widetilde{A}} \otimes \vec{\mathbf{v}}) \qquad \pi_{\widetilde{A},2} = \begin{bmatrix} 1 & T^{-\sigma_1} A^{\delta_1} \\ 1 & T^{-\sigma_2} A^{\delta_2} \end{bmatrix} \circledcirc (Z_{\widetilde{A}} \otimes \vec{\mathbf{u}})$$

$\boxed{\pi_{A^\dagger}}$ Proof for equation $\mathrm{E}_{A^\dagger}(A;\; D) \;:\; e(\underline{A},Y)\, e(\underline{A},\underline{D}) = e(K\cdot M,H)\, e(T,S)$

$$\pi_{A^\dagger,1} = \begin{bmatrix} v_{11}^{\alpha_1\delta_1} v_{21}^{\alpha_1\delta_2} & (YD)^{\alpha_1} v_{12}^{\alpha_1\delta_1} v_{22}^{\alpha_1\delta_2} \\ v_{11}^{\alpha_2\delta_1} v_{21}^{\alpha_2\delta_2} & (YD)^{\alpha_2} v_{12}^{\alpha_2\delta_1} v_{22}^{\alpha_2\delta_2} \end{bmatrix} \circledcirc (Z_{A^\dagger} \otimes \vec{\mathbf{v}}) \qquad \pi_{A^\dagger,2} = \begin{bmatrix} 1 & A^{\delta_1} \\ 1 & A^{\delta_2} \end{bmatrix} \circledcirc (Z_{A^\dagger} \otimes \vec{\mathbf{u}})$$

$\boxed{\pi_{\bar{A}}}$ Proof for equation $\mathrm{E}_{\bar{A}}(M) \;:\; e(\underline{M},H^{-1}) = e(A,Y\cdot D)^{-1} e(K,H)\, e(T,S)$

$$\pi_{\bar{A},1} = \begin{bmatrix} 1 & H^{-\mu_1} \\ 1 & H^{-\mu_2} \end{bmatrix} \circledcirc (Z_{\bar{A}} \otimes \vec{\mathbf{v}}) \qquad \pi_{\bar{A},2} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \circledcirc (Z_{\bar{A}} \otimes \vec{\mathbf{u}})$$

$\boxed{\pi_{\widehat{A}}}$ Proof for equation $\mathrm{E}_{\widehat{A}}(A,M;\; S,Y,D) \;:\; e(T^{-1},\underline{S})\, e(\underline{M},H^{-1})\, e(\underline{A},\underline{Y})e(\underline{A},\underline{D}) = e(K,H)$

$$\pi_{\widehat{A},1} := \begin{bmatrix} v_{11}^{\alpha_1(\psi_1+\delta_1)} v_{21}^{\alpha_1(\psi_2+\delta_2)} & (YD)^{\alpha_1} H^{-\mu_1} v_{12}^{\alpha_1(\psi_1+\delta_1)} v_{22}^{\alpha_1(\psi_2+\delta_2)} \\ v_{11}^{\alpha_2(\psi_1+\delta_1)} v_{21}^{\alpha_2(\psi_2+\delta_2)} & (YD)^{\alpha_2} H^{-\mu_2} v_{12}^{\alpha_2(\psi_1+\delta_1)} v_{22}^{\alpha_2(\psi_2+\delta_2)} \end{bmatrix} \circledcirc (Z_{\widehat{A}} \otimes \vec{\mathbf{v}})$$

$$\pi_{\widehat{A},2} := \begin{bmatrix} 1 & T^{-\sigma_1} A^{\psi_1+\delta_1} \\ 1 & T^{-\sigma_2} A^{\psi_2+\delta_2} \end{bmatrix} \circledcirc (Z_{\widehat{A}} \otimes \vec{\mathbf{u}})$$

Figure 5.2: Overview of different variants of the proof for the first verification equation.

5.5 Commuting Signatures on Several Messages

Automorphic Signatures on Two Messages. Applying the "pair transformation" from Sect. 4.3 (p. 41) to **Sig** (Scheme 2, p. 37) from Sect. 4.1 we get the following automorphic signature scheme[5] that signs two messages at once—if we restrict the message space to

$$\mathcal{DH}^* := \{(G^m, H^m) \,|\, m \in \mathbb{Z}_p^*\} = \mathcal{DH} \setminus \{(1,1)\} \ .$$

$\mathbf{Sig}^* := (\mathsf{Setup}_{\mathrm{S}}, \mathsf{KeyGen}_{\mathrm{S}}, \mathsf{Sign}^*, \mathsf{Ver}^*)$, where $\mathsf{Sign}^*(sk, (V,W), (M,N))$ for $(V,W), (M,N) \in \mathcal{DH}^*$ is defined as follows: pick a key pair $(vk^*, sk^*) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}$ and output[6]

$$\big(vk^*,\ \mathsf{Sign}(sk,\ vk^*),\ \mathsf{Sign}(sk^*,\ (M,N)),\ \mathsf{Sign}(sk^*,\ (V,W) \circledcirc (M,N)),\ \mathsf{Sign}(sk^*,\ (V,W)^3 \circledcirc (M,N))\big) \ .$$

$\mathsf{Ver}^*\big(vk, (V,W), (M,N), \Sigma\big)$ parses Σ as $(vk^*, \Sigma_0, \Sigma_1, \Sigma_2, \Sigma_3)$ and outputs

$$\mathsf{Ver}(vk, vk^*, \Sigma_0) \cdot \mathsf{Ver}(vk^*, (M,N), \Sigma_1) \cdot \mathsf{Ver}(vk^*, (V,W) \circledcirc (M,N), \Sigma_2) \cdot \mathsf{Ver}(vk^*, (V,W)^3 \circledcirc (M,N), \Sigma_3) \ .$$

$\mathbf{Sig}^*$ is unforgeable under ADH-SDH and AWF-CDH by Theorems 5 and 3.

Applying the vector transform (Definition 3), $\mathbf{Sig}^*$ can easily be extended for arbitrary many messages. The scheme however has message space $\mathcal{M}^* := \mathcal{DH}^*$. In this section, we define randomizable, extractable commitments to elements from $\mathcal{M}^*$, and show how to make commitments to a signature and a proof of validity on a clear and a committed message from $\mathcal{M}^*$. We define the following:

Committing to $\mathcal{M}^*$ Elements. We define $\mathbf{Com}^*_{\mathcal{M}}$ that has the same properties as $\mathbf{Com}_{\mathcal{M}}$, but with value space $\mathcal{M}^*$ rather than $\mathcal{M}$. By $\mathcal{C}^*_{\mathcal{M}}$ we denote the commitment space and by $\mathcal{R}^*_{\mathcal{M}}$ the space of randomness. Our instantiation is given in Section 5.5.1.

Partially Blind Automorphic Signatures. Since $\mathbf{Sig}^*$ signs two messages, we can define a variant of SigCom that gets one message in the clear and one committed message. Based on $\mathbf{Sig}^*$ and $\mathbf{Com}^*_{\mathcal{M}}$, we define $\mathsf{PSigCom}$ that is given a message $M \in \mathcal{M}^*$ and a $\mathbf{Com}^*_{\mathcal{M}}$ commitment and outputs a proof of knowledge of a $\mathbf{Sig}^*$ signature on M and the committed value:

$\mathsf{PSigCom}((ck, pp_S), sk, M, \mathbf{C})$. If $M \in \mathcal{M}^*$ and $\mathbf{C} \in \mathcal{C}^*_{\mathcal{M}}$ then the algorithm outputs a commitment to a signature and a proof of validity $(\mathbf{c}_\Sigma, \pi)$ which is distributed as

$$\Big[\Sigma \leftarrow \mathsf{Sign}^*(sk, (M,V));\ \rho \leftarrow \mathcal{R}\ :\ \big(\mathsf{Com}(ck, \Sigma, \rho),\ \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}^*(vk,(M,\cdot),\cdot)}, (V,\nu), (\Sigma,\rho))\big)\Big] \ ,$$

where V and ν are such that $\mathbf{C} = \mathsf{Com}^*_{\mathcal{M}}(ck, V, \nu)$.

Note that if in the construction of a blind signature in Sect. 5.2.1 (p. 50) we replace $\mathbf{Sig}$, $\mathbf{Com}_{\mathcal{M}}$ and SigCom by $\mathbf{Sig}^*$, $\mathbf{Com}^*_{\mathcal{M}}$ and $\mathsf{PSigCom}$, we obtain *partially blind signatures* [AF96] (where the signer controls part of the message), which are automorphic.

[5] In Sect. 7.3.1, we give a variant of the scheme $\mathbf{Sig}$ with messages in $\mathbb{Z}_p \times \mathcal{DH}$ (required by our application to credentials in Chapter 7) which does *not* increase the size of a signature.

[6] Exponentiation of a DH pair is defined componentwise: $(M,N)^k := (M^k, N^k)$.

5.5.1 Commitments to Non-trivial Messages

We instantiate $\mathbf{Com}^*_{\mathcal{M}}$, with message space $\mathcal{M}^* := \mathcal{DH}^* = \{(G^m, H^m) \mid m \in \mathbb{Z}_p \backslash \{0\}\}$. To guarantee that the committed value is not $(1,1)$, $\mathsf{Com}^*_{\mathcal{M}}$ must contain additional elements. Intuitively, given $(M,N) \in \mathcal{M}^*$ if we choose $l \leftarrow \mathbb{Z}_p^*$ and publish $W = N^l$ then $W \neq 1$ only if $N \neq 1$. We add a commitment $\mathbf{c}_L$ to G^l and a proof π_W that $e(G^l, N) = e(G, W)$, which proves well-formedness of W. In the WI setting $W, \mathbf{c}_L, \mathbf{c}_N$ and π_W perfectly hide N.

Randomization of a $\mathbf{Com}^*_{\mathcal{M}}$ commitment is a bit trickier than for $\mathbf{Com}_{\mathcal{M}}$ commitments. Since l must be from $\mathbb{Z}_p^*$, we randomize it *multiplicatively*, that is, we choose $l' \leftarrow \mathbb{Z}_p^*$ and replace l by $l \cdot l'$, which is then again in $\mathbb{Z}_p^*$. This also enables randomization of W as $W' := W^{l'}$ which without knowledge of N cannot be done additively. (The component U of a $\mathbf{Com}_{\mathcal{M}}$ commitment (p. 54) could be randomized additively, since $U = T^t \cdot M$ and T is public.) Finally, Lemma 6 (p. 53) shows how to adapt $\mathbf{c}_L$ and π_W to the new value $l \cdot l'$.

We define $\mathbf{Com}^*_{\mathcal{M}}$ by extending $\mathbf{Com}_{\mathcal{M}}$ from Sect. 5.4.1 (p. 54):

$\mathsf{Com}^*_{\mathcal{M}}(pp, (M,N), (\kappa = (t, \mu, \nu, \rho, \sigma), \eta, l))$. If $(M,N) \in \mathcal{DH}^*$, $(\kappa, \eta, l) \in \mathcal{R}_{\mathcal{M}} \times \mathcal{R} \times \mathbb{Z}_p^* =: \mathcal{R}^*_{\mathcal{M}}$ then define

$$W := N^l \qquad \mathbf{c}_L := \mathsf{Com}(ck, G^l, \eta) \qquad \pi_W \leftarrow \mathsf{Prove}(ck, \mathrm{E}_W, (G^l, \eta), (N, \nu))$$

with $\mathrm{E}_W(L;\, N) :\ e(\underline{L}, \underline{N}) = e(G, W)$, and output $\big(\mathsf{Com}_{\mathcal{M}}(ck, (M,N), \kappa), W, \mathbf{c}_L, \pi_W\big)$.

The space of valid commitments is

$$\mathcal{C}^*_{\mathcal{M}} := \big\{(\mathbf{C}, W, \mathbf{c}_L, \pi_W) \mid \mathbf{C} \in \mathcal{C}_{\mathcal{M}} \ \wedge\ W \neq 1 \ \wedge\ \mathsf{Verify}(ck, \mathrm{E}_W, \mathbf{c}_L, \mathbf{c}_N, \pi_W)\big\}\ .$$

$\mathbf{Com}^*_{\mathcal{M}}$ is shown to be binding and computationally hiding analogously to $\mathbf{Com}_{\mathcal{M}}$. In particular, if ck^* is output by $\mathsf{WISetup}$ then for every $(M,N) \in \mathcal{DH}^*$, given $(\mathbf{C}, W, \mathbf{c}_L, \pi_W)$, there exists (κ, η, l) such that $\mathbf{C} = \mathsf{Com}_{\mathcal{M}}(ck^*, (M,N), \kappa)$, $W = N^l$, $\mathbf{c}_L = \mathsf{Com}(ck^*, G^l, \eta)$ and $\pi_W \leftarrow \mathsf{Prove}(ck, \mathrm{E}_W, (G^l, \eta), (N, \nu))$ with $\kappa = (t, \mu, \nu, \rho, \sigma)$.

Using the results from Sect. 5.3 on properties of Groth-Sahai proofs, we define $\mathsf{RdCom}^*_{\mathcal{M}}$ by extending $\mathsf{RdCom}_{\mathcal{M}}$:

$\mathsf{RdCom}^*_{\mathcal{M}}(pp, (\mathbf{C}, W, \mathbf{c}_L, \pi_W), (\kappa', \eta', l'))$ returns a commitment $(\mathbf{C}', W', \mathbf{c}'_L, \pi'_W)$ that is equivalent to the output of $\mathsf{Com}^*_{\mathcal{M}}(pp, (M,N), (\kappa + \kappa', l' \cdot \eta + \eta', l \cdot l'))$ computed as $\mathbf{C}' := \mathsf{RdCom}_{\mathcal{M}}(pp, \mathbf{C}, \kappa')$ and

$$W' := W^{l'} \quad \mathbf{c}'_L := \mathsf{RdCom}(ck, (\mathbf{c}_L^{l'}), \eta') \quad \pi'_W \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_W, (\mathbf{c}_L^{l'}, \eta'), (\mathbf{c}_N, \nu'), \pi_W^{l'})\big)$$

$\mathsf{RdCom}^*_{\mathcal{M}}$ works as follows: the part $\mathbf{C}$ of a commitment is randomized by $\mathsf{RdCom}_{\mathcal{M}}$ which replaces κ by $\kappa + \kappa'$. Now W is replaced by $W^{l'}$, which implicitly replaces l by $l \cdot l'$. Setting $\hat{\mathbf{c}}_L := \mathbf{c}_L^{l'}$, we get $\hat{\mathbf{c}}_L = \mathsf{Com}(ck, L^{l'}, l' \cdot \eta)$ and by Lemma 6, we have that $\hat{\pi}_W := \pi_W^{l'}$ is a proof for E_W and $(\hat{\mathbf{c}}_L, \mathbf{c}_N)$. In $W', \hat{\mathbf{c}}_L$ and $\hat{\pi}_W$, randomness l was thus consistently replaced by $l \cdot l'$. The final step is to set $\mathbf{c}'_L = \mathsf{RdCom}(ck, \hat{\mathbf{c}}_L, \eta')$ and $\pi'_W \leftarrow \mathsf{RdProof}(ck, \mathrm{E}_W, (\hat{\mathbf{c}}_L, \eta'), (\mathbf{c}_N, \nu'), \hat{\pi}_W)$. Note that $\mathbf{c}'_L$ is thus a commitment to $L^{l'}$ under randomness $l' \cdot \eta + \eta'$.

If (κ', η', l') is uniformly chosen from $\mathcal{R}^*_{\mathcal{M}}$ then the randomness after randomization is also uniform in $\mathcal{R}^*_{\mathcal{M}}$.

5.5.2 Making Commitments to a Signature on a Public and a Committed Message and a Proof of Validity

We now give an instantiation of $\mathsf{PSigCom}$ defined at the beginning of the section. We start by giving a variant of SigCom that has inputs $(ck, sk, (V, W), \mathbf{C})$ with $(V, W) \in \mathcal{M}^*$ and $\mathbf{C} \in \mathcal{C}^*_{\mathcal{M}}$ and outputs a proof of knowledge of a signature on $(V, W) \circledcirc (M, N)$, where (M, N) is the message committed in $\mathbf{C}$. The verification of a signature on such a product $\mathsf{Ver}'((X, Y), (V, W), (M, N), (A, B, D, R, S))$ is done by checking the following equations: $\mathrm{E}_{A'}$, defined as

$$\mathrm{E}_{A'}(A, M;\, S, D) \;:\; e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K \cdot V, H) \;\;,$$

and E_B, E_R as defined in (5.5) on p. 56. Since the left-hand sides of $\mathrm{E}_{A'}$ and E_A from (5.5) are equivalent, by Lemma 3 both equations have the same proofs: $\pi_A = \pi_{A'}$. The *only* thing that changes w.r.t. SigCom is thus the value A of the pre-signature.

$\mathsf{SigCom}'(ck, x, (V, W), \mathbf{C})$. This variant is defined as SigCom in Figure 5.1, except that A is defined as $A := (K \cdot T^r \cdot U \cdot V)^{\frac{1}{x+c}}$.

Proof adaptation to a committed verification key by $\mathsf{AdPrC}_{\mathcal{K}}$ can also be applied to outputs of SigCom', since proofs only depend on the left-hand sides of the equations they prove. Let $\mathrm{E}_{\widehat{A}'}$ be $\mathrm{E}_{A'}$ as defined above but with Y being a variable. Since $\mathrm{E}_{A'}$ and E_A, as well as $\mathrm{E}_{\widehat{A}}$ and $\mathrm{E}_{\widehat{A}'}$, have the same left-hand sides, $\mathsf{AdPrC}_{\mathcal{K}}$, which transforms a proof for E_A into a proof for $\mathrm{E}_{\widehat{A}}$, also transforms a proof for $\mathrm{E}_{A'}$ into one for $\mathrm{E}_{\widehat{A}'}$.

$\mathsf{PSigCom}$ first creates a temporary key pair (vk', sk'), signs vk' with the secret key sk, commits to vk' and the signature and proves validity. It then uses SigCom' to create commitments to signatures on the linear combinations of the clear message (V, W) and the message committed in $\mathbf{C}$. The output is a commitment to a signature consisting of vk', the signature Σ_0 on vk' and the signatures Σ_1, Σ_2 and Σ_3 on the linear combinations of the messages.

$\mathsf{PSigCom}(ck, sk, (V, W), \mathbf{C})$.

- $((X', Y'), x') \leftarrow \mathsf{KeyGen}_{\mathrm{S}};\ (\xi', \psi') \leftarrow \mathcal{R}^2$
  $\mathbf{c}_X := \mathsf{Com}(ck, X', \xi');\ \mathbf{c}_Y := \mathsf{Com}(ck, Y', \psi');\ \pi_X := \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (X', \xi'), (Y', \psi'))$
- $\Sigma_0 \leftarrow \mathsf{Sign}(sk, (X', Y'));\ \sigma_0 \leftarrow \mathcal{R}^5$
  $\mathbf{c}_{\Sigma_0} := \mathsf{Com}(ck, \Sigma_0, \sigma_0);\ \pi_0 \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathsf{Ver}(vk, \cdot, \cdot)}, (X', \xi'), (Y', \psi'), (\Sigma_0, \sigma_0))$
- $(\mathbf{c}_{\Sigma_1}, \pi'_1) \leftarrow \mathsf{SigCom}(ck, x', \mathbf{C});\ \pi_1 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, ((X', Y'), (\xi', \psi')), \mathbf{C}, \mathbf{c}_{\Sigma_1}, \pi'_1)$
  $(\mathbf{c}_{\Sigma_2}, \pi'_2) \leftarrow \mathsf{SigCom}'(ck, x', (V, W), \mathbf{C});\ \pi_2 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, ((X', Y'), (\xi', \psi')), \mathbf{C}, \mathbf{c}_{\Sigma_2}, \pi'_2)$
  $(\mathbf{c}_{\Sigma_3}, \pi'_3) \leftarrow \mathsf{SigCom}'(ck, x', (V, W)^3, \mathbf{C});\ \pi_3 \leftarrow \mathsf{AdPrC}_{\mathcal{K}}(ck, ((X', Y'), (\xi', \psi')), \mathbf{C}, \mathbf{c}_{\Sigma_3}, \pi'_3)$
- Return $\big(\mathbf{c}_\Sigma = (\mathbf{c}_{\Sigma_0}, \mathbf{c}_X, \mathbf{c}_Y, \mathbf{c}_{\Sigma_1}, \mathbf{c}_{\Sigma_2}, \mathbf{c}_{\Sigma_3}), \pi = (\pi_0, \pi_X, \pi_1, \pi_2, \pi_3)\big)$

A proof of knowledge of a signature $(\mathbf{c}_\Sigma, \pi)$ under vk on the message pair $(V, W) \in \mathcal{M}^*$ and (M, N), which is given as a commitment $\mathbf{C} \in \mathcal{C}^*_{\mathcal{M}}$ is then verified by checking the following:

$$\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(vk, \cdot, \cdot)}, (\mathbf{c}_X, \mathbf{c}_Y), \mathbf{c}_{\Sigma_0}, \pi_0),\ \mathsf{Verify}(ck, \mathrm{E}_{\mathcal{DH}}, \mathbf{c}_X, \mathbf{c}_Y, \pi_X),$$
$$\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}(\cdot, \cdot, \cdot)}, (\mathbf{c}_X, \mathbf{c}_Y), \mathbf{C}, \mathbf{c}_{\Sigma_1}, \pi_1),\ \mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}'(\cdot, (V,W), \cdot, \cdot)}, (\mathbf{c}_X, \mathbf{c}_Y), \mathbf{C}, \mathbf{c}_{\Sigma_2}, \pi_2),$$
$$\mathsf{Verify}(ck, \mathrm{E}_{\mathsf{Ver}'(\cdot, (V,W)^3, \cdot, \cdot)}, (\mathbf{c}_X, \mathbf{c}_Y), \mathbf{C}, \mathbf{c}_{\Sigma_3}, \pi_3) \;\;.$$

$\mathsf{AdPrC}^*_{\mathcal{K}}\big(ck,\ ((X,Y),(\xi,\psi)),\ (V,W),\mathbf{C},(\mathbf{c}_{\Sigma_0},\mathbf{c}_{X'},\mathbf{c}_{Y'},\mathbf{c}_{\Sigma_1},\mathbf{c}_{\Sigma_2},\mathbf{c}_{\Sigma_3}),(\pi_0,\pi_X,\pi_1,\pi_2,\pi_3)\big)$ transforms a proof of validity of a signature committed in $\mathbf{c}_\Sigma$ on (V,W) and the content of $\mathbf{C}$ into a proof for when the verification key (X,Y) is committed as well.

Let $\mathbf{c}_{\Sigma_0} = (\mathbf{c}_{A_0},\mathbf{c}_{B_0},\mathbf{c}_{D_0},\mathbf{c}_{R_0},\mathbf{c}_{S_0})$ and $\pi_0 = (\pi_{0_A},\pi_{0_B},\pi_{0_R})$. Analogously to $\mathsf{AdPrC}_{\mathcal{K}}$ (p. 59), $\mathsf{AdPrC}^*_{\mathcal{K}}$ sets

$$\widehat{\pi}_{0_A} \leftarrow \mathsf{RdProof}\big(ck,\mathrm{E}_{\widehat{A}},(\mathbf{c}_{0_A},0),(\mathbf{c}_{X'},0),(\mathbf{c}_{0_S},0),(\mathsf{Com}(ck,Y,0),\psi),(\mathbf{c}_{0_D},0),\pi_{0_A}\big)$$

and outputs a randomization of $\widehat{\pi} := \big((\widehat{\pi}_{0_A},\pi_{0_B},\pi_{0_R}),\pi_X,\pi_1,\pi_2,\pi_3\big)$.

Chapter 6

Applications of Automorphic Signatures

Contents

In this chapter we describe applications of automorphic signatures, which we defined and instantiated in Chapter 4. Combining them with Groth-Sahai proofs (Sect. 2.4) we show how to instantiate the following primitives in a black-box way: the round-optimal blind signatures by Fischlin [Fis06]; the non-interactive anonymous credentials by Belenkiy et al. [BCKL08]; and group signatures satisfying the BSZ model, for which we use the construction by Groth [Gro07].

The second part of the chapter is dedicated to anonymous proxy signatures. We start by presenting the model that we defined in [1] and give the first efficient instantiation, which makes use of the full power of automorphic signatures. We then show how to achieve stronger anonymity definitions and conclude by giving an instantiation in detail. The results of this chapter appear in [6].

6.1 Black-Box Applications of Automorphic Signatures

6.1.1 Round-Optimal Blind Signatures

In [Fis06], Fischlin gives a generic construction for concurrently executable blind-signature schemes with optimal round complexity in the common reference string (CRS) model. The construction relies on commitment, encryption and signature schemes and generic NIZK proofs for NP-languages. In the signature-issuing protocol, the user first sends a commitment to the message to the signer (issuer), who responds with a signature on the commitment. The user then constructs the blind signature as follows: she encrypts the commitment and the signature and adds a NIZK proof that the signature is valid on the commitment and that the committed value is the message.

Following [HKKL07], Abe and Ohkubo [AO09] replace the NIZK proof in Fischlin's construction by a witness-indistinguishable proof and concretely suggest Groth-Sahai (GS) proofs. (Note that GS commitments on group elements can be "decrypted" using the extraction key.) To be compatible, the signature scheme must have messages and signatures consisting of group elements and verification must amount to evaluating pairing-product equations. However, they only mention the highly inefficient scheme from [Gro06] as a feasibility result and leave open the problem of an efficient construction. Automorphic signatures satisfy all the compatibility requirements and enable thus an efficient instantiation of round-optimal blind signatures; it suffices to construct a commitment scheme such that commitments lie in the message space of the signature (or are vectors of messages) and correct opening is verifiable by PPEs (as has been done in [AHO10] since).

We directly constructed a scheme in Sect. 4.2 which has smaller blind signatures than an instantiation of the generic construction: in the end of our issuing protocol, the user holds a signature on the *message* rather than on a commitment to it. To make this possible, the user sends a *randomization* of the message to the issuer in addition to the commitment. From this, the issuer makes a "pre-signature" and sends it to the user, who turns it into an actual signature on the message by adapting the randomness. The blind signature is then a GS proof of knowledge of a signature on the message (rather than a commitment), which avoids encrypting (committing to) the commitment and proving that the commitment opens to the message. The size of our signature is around 30 group elements (depending on the GS instantiation) and the two messages sent during issuing are even smaller.

6.1.2 P-Signatures and Anonymous Credentials

In order to realize *non-interactive anonymous credentials*, Belenkiy et al. [BCKL08] introduce a new primitive called *P-signature*. It extends a signature and a commitment scheme by the following functionalities: an interactive protocol $\mathsf{Issue} \leftrightarrow \mathsf{Obtain}$ between a signer and a user allows the latter to obtain a signature on a value the signer only knows a commitment to; and the holder of a message and a signature on it can produce a commitment to the message and a proof of knowledge of the signature. The commitments and proofs are instantiated with the Groth-Sahai methodology; the compatible signature scheme is the one discussed in Sect. 1.1. Using an automorphic signature instead has the following advantages: the signatures and messages being group elements, they can be extracted in the security reduction, which avoids notions like F-unforgeability. Moreover, a small modification of the signature-issuing protocol of our blind signatures (cf. Remark 4, p. 41) yields an efficient $\mathsf{Issue} \leftrightarrow \mathsf{Obtain}$ protocol (whereas the one in [BCKL08] resorts to generic secure multiparty computation).

6.1.3 Fully-Secure Group Signatures

In order to implement the model for group signatures by [BSZ05], Groth [Gro07] uses the following ingredients to achieve CCA-anonymity:[1] a *strong one-time signature scheme*[2] $\mathbf{Sig}_{\mathrm{ot}}$ and the *tag-based encryption scheme* [MRY04] $\mathbf{Enc}_{\mathrm{tb}}$ by Kiltz [Kil06]. A tag-based encryption scheme is a public-key encryption scheme whose encryption and decryption algorithms take as additional

[1] A group-signature scheme is *CCA-anonymous* if no adversary can decide which of two users created a group signature, even if he can query opening of any other group signature.

[2] More precisely, the scheme must be strongly unforgeable against weak one-time chosen-message attacks, which means that if an adversary makes a single chosen-message query *before receiving the public key* it can neither output a new signed message nor a new signature on the queried message. The signatures Groth uses are the weak Boneh-Boyen signatures from [BB04].

argument a *tag*. Kiltz' scheme is *selective-tag weakly CCA-secure*, i.e., an adversary outputting a tag t^* (before receiving the public key) and two messages and getting an encryption of one of them under t^* cannot decide which one was encrypted—even when provided with an oracle decrypting any ciphertext with tag $t \neq t^*$.

In Groth's group-signature scheme a user produces a signature key pair (vk, sk) and is enrolled by the issuer who gives her a certificate *cert* on *vk*. Now to make a group signature on a message M, the user holding $(cert, vk, sk)$ generates a key pair $(vk_{\mathrm{ot}}, sk_{\mathrm{ot}})$ for $\mathbf{Sig}_{\mathrm{ot}}$, makes a signature *sig* on vk_{ot} under *vk* and produces a Groth-Sahai WI proof of knowledge ξ of $(cert, vk, sig)$ s.t. *cert* is a valid certificate on *vk* and *sig* is a signature on vk_{ot} valid under *vk*. She produces an $\mathbf{Enc}_{\mathrm{tb}}$-ciphertext C encrypting *sig* under tag vk_{ot} and adds a Groth-Sahai NIZK proof ζ that the encrypted value *sig* is the same as in the commitment contained in ξ. Using sk_{ot}, she finally makes a signature sig_{ot} on $(M, vk_{\mathrm{ot}}, \xi, C, \zeta)$ and outputs the group signature $\sigma = (vk_{\mathrm{ot}}, \xi, C, \zeta, sig_{\mathrm{ot}})$. To verify σ, check whether sig_{ot}, the proofs ξ and ζ, and the ciphertext C are valid. The opener holds a key enabling her to extract $(cert, vk, sig)$ from ξ. The key *vk* allows to determine the signer and *sig* acts as a non-frameable proof of correct tracing.

Using automorphic signatures to instantiate the schemes for *cert* and *sig* immediately yields a group signature scheme secure in the BSZ-model. More concretely, in [4] we suggested to substitute the certified-signature scheme used by Groth, which is based on the "q-U Assumption", by one based on the more natural DH-SDH (Sect. 3.2). Our replacement however used Waters signatures [Wat05] which entail a dramatic increase of the public-key size. This can be avoided by using instead the certified-signature scheme given in Remark 2 (p. 36), which is based on DH-SDH as well.

6.2 Anonymous Proxy Signatures

6.2.1 The Model

Anonymous proxy signatures (APS) generalize group signatures in that everyone can become a group manager by delegating his signing rights to other users who can then anonymously sign in his name; moreover, received rights can be *re-delegated*. To make it possible that an opening authority can trace the anonymous delegators and signers, the users have to *register* with an issuing authority when joining the system. We give a brief overview of our model defined in [1].

Algorithm Setup establishes the public parameters and also outputs a key for the issuer. Users generate key pairs using KeyGen and run a protocol Reg with the *issuer* and their *opener* when joining the system. To delegate to Bob, Alice runs Delgt on Bob's public key, which produces a *warrant* she gives to Bob. With this warrant, Bob can either sign or *re-delegate* to Carol, in which case Carol can again re-delegate or produce a *proxy signature* with PSign on behalf of Alice, which is verifiable by Ver on Alice's verification key.

Anonymity ensures that from a proxy signature one cannot tell who actually signed (or re-delegated), thus Bob and Carol remain anonymous. Anonymity is formalized as follows: The adversary gets the parameters and the issuing key (and can therefore register and delegate users); it then outputs: a public key (of the original delegator), two secret keys and warrants for them and a message. If the warrants are both valid and of equal length, the adversary is given a proxy signature using one of them and has to decide which one.

In case of dispute, Alice's opener can revoke the anonymity of the intermediate delegators and the proxy signer: given a proxy signature it outputs a list of users representing the delegation chain, and a proof of correctness of opening. The notion of unforgeability is defined analogously to that of

group signatures in [BSZ05]. *Traceability* asserts that every valid proxy signature can be opened to registered users and *non-frameability* guarantees that no adversary, even when colluding with the issuer, openers and other users, can produce a signature, a list of users and a proof that accuses an honest user of a delegation or a signing she did not perform.

A Generic Construction. Our generic construction in [1] proving feasibility of the model is as follows. Assume an EUF-CMA-secure signature scheme, an encryption scheme with indistinguishable ciphertexts under chosen-ciphertext attack [RS92], and a simulation-sound zero-knowledge proof system for NP languages [Sah99].

Setup chooses a key pair for the signature scheme and a common reference string (CRS) for the NIZK proof system. The verification key and the CRS are defined as the public parameters and the signing key is given to the issuer. When enrolling, a user U_i chooses a signing/verification key pair and obtains a signature $cert_i$ on her verification key vk_i from the issuer. U_i's opener chooses a key pair for the encryption scheme, of which the user adds the encryption key to her public key pk_i and the opener keeps the decryption key as opening key.

A warrant $warr_{1\to 2}$ from user U_1 to user U_2 is a signature on (vk_1, vk_2) valid under vk_1. U_2 re-delegates to U_3 by sending $warr_{1\to 2}$ and $warr_{2\to 3}$, a signature on (vk_1, vk_2, vk_3) under vk_2. Additionally, in each delegation step, the delegators' certificates are also passed on. Given a warrant $(warr_{1\to 2}, warr_{2\to 3})$, U_3 proxy-signs a message M on behalf of U_1 as follows: first produce a signature sig on (vk_1, vk_2, vk_3, M) using sk_3; then define the *plain* proxy signature as $(warr_{1\to 2}, vk_2, cert_2, warr_{2\to 3}, vk_3, cert_3, sig)$. In general we say that a plain proxy signature $\Sigma = (warr_{1\to 2}, \ldots, vk_k, cert_k, sig)$ on message M is valid under key vk_1 if:

- $\forall i$: $cert_i$ is a signature on vk_i valid under the issuer's verification key;
- $\forall i$: $warr_{i\to i+1}$ is a signature on $(vk_1, \ldots, vk_{i+1})$ valid under vk_i; ($\mathrm{Ver}_{\mathrm{PPS}}$)
- sig is a signature on $(vk_1, \ldots, vk_k, M)$ valid under vk_k.

Now to transform this into an *anonymous* proxy signature, the signer encrypts Σ under the public key of U_1's opener (contained in vk_1) and adds a NIZK proof that the plaintext satisfies the verification relations of a plain proxy signature in ($\mathrm{Ver}_{\mathrm{PPS}}$). Due to her decryption key, the opener can retrieve the plain signature and thus trace the delegators and the signer. The warrants and sig are non-frameable proofs of correct tracing.

6.2.2 Concrete Instantiations

Restricting the model to CPA-anonymity (where the adversary trying to break anonymity is not given access to an opening oracle), the building blocks can be instantiated as follows: define encryption to be Groth-Sahai (GS) commitments (which can be "decrypted" due to extractability) and use GS proofs to show that the verification relations are satisfied by the committed values. For this to work however, the plain proxy signatures must fit the GS framework; meaning that the EUF-CMA signature scheme's verification keys, messages and signatures must be group elements satisfying pairing-product equations, and the signing keys must lie in the message space; in short, they must be automorphic signatures. In [3] we gave a CPA-anonymous instantiation of APS which is however fairly inefficient due to the used signature scheme (its public keys contain several commitments to each *bit* of the corresponding secret key). Moreover, we only considered one general opener and there is a maximum number of consecutive re-delegations. These limitations are easily overcome by using automorphic signatures.

In the next section, we show how to make the above scheme CCA-anonymous and thus fully satisfy the security model defined in [1]. In Sect. 6.2.4 we discuss how to sign one message on

behalf of several delegators and in Sect. 6.2.5 we show how to achieve anonymity of the delegator or the delegatee w.r.t. each other. (The model in [1] only considers anonymity towards the verifier.) In all our constructions, public *attributes* can be easily included as messages for the signatures in delegation. The delegators can thus specify for which tasks they delegate signing rights.

6.2.3 CCA-Anonymous Proxy Signatures

CCA-anonymity (i.e., anonymity against adversaries provided with an opening oracle) of Groth's group signatures [Gro07] (sketched in Sect. 6.1.3, p. 67) is proved as follows: modify the security game by substituting the opener's commitment key by one that results in perfectly hiding commitments and perfectly WI proofs; due to the additional encryptions contained in a group signature, opening queries for all but the challenge signature can still be simulated. So a break of anonymity can be used to break the security of the tag-based encryption scheme.

We transform the anonymous proxy signature scheme given in the previous section into one satisfying CCA-anonymity analogously. Let $\mathbf{Sig}_{\mathrm{ot}}$ denote the weak Boneh-Boyen signatures from [BB04] and let $\mathbf{Enc}_{\mathrm{tb}}$ denote Kiltz' tag-based encryption scheme from [Kil06]. Suppose a proxy signer holds $W := (vk_1, (warr_i, cert_i, vk_i)_{i=2}^k)$ and sk_k. To make a signature, she first chooses keys $(vk_{\mathrm{ot}}, sk_{\mathrm{ot}}) \leftarrow \mathsf{KeyGen}_{\mathrm{ot}}$ and signs vk_{ot} (instead of M) with her personal key sk_k yielding sig. She makes commitments $\vec{\mathbf{c}}$ to the elements of W and sig, and adds a WI proof π_j for each equation E_j in ($\mathrm{Ver}_{\mathrm{PPS}}$) in Sect. 6.2.1, which are satisfied by W and sig—as in the original scheme.

In addition, for $2 \le i \le k$ she computes an $\mathbf{Enc}_{\mathrm{tb}}$-encryption C_i of $warr_i$ under tag vk_{ot} and, as in [Gro07], she makes a Groth-Sahai NIZK proof ζ_i that the plaintext of C_i is the value committed in $\mathbf{c}_{warr_i}$. She computes $sig_{\mathrm{ot}} := \mathsf{Sign}_{\mathrm{ot}}(sk_{\mathrm{ot}}, (vk_{\mathrm{ot}}, M, \vec{\mathbf{c}}, \vec{\pi}, \vec{C}, \vec{\zeta}))$ and outputs the signature $(vk_{\mathrm{ot}}, \vec{\mathbf{c}}, \vec{\pi}, \vec{C}, \vec{\zeta}, sig_{\mathrm{ot}})$. A signature is *valid* if sig_{ot} is valid under vk_{ot}, the proofs π_j are valid for all j, and the proofs ζ_i and the ciphertexts C_i are valid for all i. Given a valid signature, the opener returns the values $(vk_i, warr_i)_{i=1}^k$ extracted from the commitments $\vec{\mathbf{c}}$ using the extraction key.

The proof for CCA-anonymity is analogous to that for Groth's group signatures. Let Game 0 denote the game for CCA-anonymity defined in [1]. The adversary $\mathcal{A}$ controls the issuer and the users and has on opening oracle simulating an honest opener. After the first phase $\mathcal{A}$ returns a public key pk for an original delegator, two user secret keys and two valid warrants of equal length from pk to the users, as well as a message. $\mathcal{A}$ receives an anonymous proxy signature produced with one of the secret keys and the corresponding warrant. After a second phase of opening queries, $\mathcal{A}$ has to decide which key/warrant pair was used.

In Game 1, the opening queries are simulated by decrypting $\vec{C}$, checking for which users the warrants are valid and returning their registered keys together with the warrants. Soundness of the proofs $\vec{\zeta}$ guarantees perfect simulation. In Game 2, we replace the opener's commitment key by a witness-indistinguishable key and in Game 3 we simulate the proofs in $\vec{\zeta}$. The unforgeability of the one-time signature $\mathbf{Sig}_{\mathrm{ot}}$ prevents the adversary from querying opening of a proxy signature which is different from the challenge but contains the same vk_{ot}. We can thus use an adversary winning Game 3 to break selective-tag weak CCA security of $\mathbf{Enc}_{\mathrm{tb}}$ (see Sect. 6.1.3, p. 67) since we only have to answer decryption queries for tags $vk'_{\mathrm{ot}} \neq vk_{\mathrm{ot}}$.

6.2.4 Multiple Original Delegators

If in anonymous proxy signatures, we allow delegation to take the form of a tree (whose leaves represent original delegators, and delegation goes from the leaves to the root) rather than a list, we can define proxy signatures on behalf of several originators. For example, consider three original

delegators O, P, Q, the first of which delegates to A who re-delegates to B. User B is also delegated by P and re-delegates the rights for both O and P to C. Moreover Q delegates to C. Now C can produce a signature on behalf of O, P *and* Q.

In general, we define a *multi-originator signature* (MOS) recursively (for simplicity we assume the verification keys contain their certificates): A (plain) MOS consists of a signature on the message, the proxy signer's verification key and a list of objects *del* for the signer (which represent the delegations to her). A *del* for user U is either a warrant from an originator for U or a warrant from a user U', the verification key of U' and a list of *del*'s for U'. A (plain) signature on behalf of a set of originators is valid if the signature on the message is valid, all warrants are valid and it contains a warrant from each of the originators. As for the single-originator case, a plain signature is anonymized by committing to its components and adding proofs of validity.

In the above example, a signature by C on behalf of O, P and Q has the following form (we let $\xi_{U_1 \to U_2}$ denote $\mathbf{c}_{U_1 \to U_2} \| \pi_{U_1 \to U_2}$, and ξ_M denote a commitment to *sig* and a proof of validity):

$$\Big\{\xi_M, \mathbf{c}_C, \big\{\big\{\xi_{B\to C}, \mathbf{c}_B, \big\{\{\xi_{A\to B}, \mathbf{c}_A, \xi_{O\to A}\}, \xi_{P\to B}\big\}\big\}, \xi_{Q\to C}\big\}\Big\} \ .$$

6.2.5 Anonymous Proxy Signatures with Enhanced Anonymity Guarantees

We briefly sketch how to instantiate the extended model of APS discussed at the end of Sect. 1.1. A formal description can be found in the end of this section.

Blind Delegation. Using our blind automorphic signatures from Sect. 4.2, we can define *blind delegation*: instead of directly signing the delegatee's public key, the delegator runs a blind issuing protocol with the delegatee. In the end, the latter holds an actual warrant and continues as in the original APS scheme. The identity of the delegatee remains thus hidden to the delegator.

Delegator Anonymity. Due to the modularity of Groth-Sahai proofs (for each equation its proof only depends on the commitments to the variables appearing in it), the "anonymization" of a signature need not be delayed until the proxy signing: warrants can be anonymized by the delegators already and randomized in each delegation step (which prevents linkability of signatures and delegations). However, we need to revise the way warrants are defined, since the present scheme requires knowledge of the identities of all previous delegators to construct them. We follow the general approach by [BCC+09], who associate an identifier *id* to each original delegation. A warrant from the user at level i in the delegation chain to the next one is then a signature on $(\mathcal{H}(id\|i), vk_{i+1})$ under vk_i, where $\mathcal{H}\colon \{0,1\}^* \to \mathbb{G}$ is a collision-resistant hash function.[3] The hash value prevents combining different warrants and reordering within a warrant.

Consider the following situation (again we simplify our exposition by assuming the certificate from the issuer is contained in the user public key, and by omitting the hash values): Oliver (the **o**riginal delegator), owning vk_O, delegated to Alice by giving her a signature $warr_{O\to A}$ on her key vk_A. Alice re-delegates to Bob sending him $(warr_{O\to A}, vk_A, warr_{A\to B})$. Bob can now delegate to Carol *without revealing Alice's identity*: He makes commitments $\mathbf{c}_{O\to A}$, $\mathbf{c}_A$ and $\mathbf{c}_{A\to B}$ to $warr_{O\to A}$, vk_A and $warr_{A\to B}$, respectively. He makes a *trivial* commitment $\mathbf{c}_B = \mathsf{Com}(ck, vk_B, 0)$ to his own key, and the following proofs: $\pi_{O\to A}$ for $\mathbf{c}_{O\to A}$ containing a valid warrant from vk_O to the content of $\mathbf{c}_A$, and $\pi_{A\to B}$ for $\mathbf{c}_{A\to B}$ containing a valid warrant from the content of $\mathbf{c}_A$ to the content of $\mathbf{c}_B$. He sends $\widetilde{warr} := (vk_O, \mathbf{c}_A, \mathbf{c}_{O\to A}, \pi_{O\to A}, \mathbf{c}_B, \mathbf{c}_{A\to B}, \pi_{A\to B})$ and a warrant $warr_{B\to C}$ to Carol.

[3] Since *id* and i are publicly known, $\mathcal{H}(id\|i) \in \mathbb{G}$ will be considered a constant in the Groth-Sahai proofs.

Now, Carol produces a signature on behalf of Oliver on M as follows (re-delegation works analogously): make a signature *sig* on M valid under vk_C; *randomize* the commitments and adapt the proofs in $\widetilde{warr}$, in particular, set $\mathbf{c}'_B := \mathsf{RdCom}(ck, \mathbf{c}_B, \rho_B)$; make commitments to $warr_{B\to C}$, vk_C and *sig*, and proofs of validity of $warr_{B\to C}$ and *sig*. Note that for the first proof the randomness of the related commitments—in particular $\mathbf{c}'_B$—is required. Since $\mathbf{c}_B$ was a trivial commitment, the randomness of $\mathbf{c}'_B$ is ρ_B which was chosen by Carol (cf. Sect. 2.4).

Remark 8. (1) Note that delegator-anonymous delegation is compatible with blind delegation: instead of simply sending $warr_{B\to C}$, Bob runs the interactive blind-issuing protocol with Carol, upon which she obtains $warr_{B\to C}$ and continues as above.

(2) Bob could also hide *his own identity* to Carol as follows: he sends (hiding) commitments to his own key and to $warr_{B\to C}$, and in addition a trivial commitment to Carol's key and proof of validity of $warr_{B\to C}$. Carol randomizes what Bob sent her, commits to a signature on the message and proves validity. In the next section, we formally describe an instantiation of anonymous proxy signatures with delegator anonymity.

6.2.6 An Anonymous Proxy Signature Scheme with Delegator Anonymity

We formally describe an instantiation of anonymous proxy signatures with delegator anonymity as discussed in Remark 8 (2).

Building Blocks. To instantiate APS with delegator anonymity, we will use the following building blocks that were introduced in Sections 2.4 and 4.1, respectively. We can instantiate them over asymmetric bilinear groups in which SXDH holds, or over symmetric groups in which DLIN is hard.

- Commitments: $\mathsf{ExSetup}(\cdot)$ takes as input the asymmetric (or symmetric) bilinear group and outputs a commitment key $ck \in \mathbb{G}_1^3 \times \mathbb{G}_2^3$ (or $ck \in \mathbb{G}^5$) and an extraction key $ek \in \mathbb{Z}_p^2$. On inputs a commitment key, a group element, and randomness from $\mathcal{R} := \mathbb{Z}_p^2$ (or $\mathcal{R} := \mathbb{Z}_p^3$), $\mathsf{Com}(\cdot,\cdot,\cdot)$ outputs a commitment consisting of 2 (or 3) group elements. $\mathsf{RdCom}(\cdot,\cdot,\cdot)$ takes a commitment key, a commitment and fresh randomness, and outputs a randomized commitment to the same value; $\mathsf{Extr}(\cdot,\cdot)$ outputs the committed value on input *ek* and a commitment.
- Groth-Sahai proofs: $\mathsf{Prove}(\cdot,\cdot,\cdot)$ produces a proof in $\mathbb{G}_1^4 \times \mathbb{G}_2^4$ (for the DLIN instantiation, the proofs are in $\mathbb{G}^3$ for linear equations, and in $\mathbb{G}^9$ for general equations) on inputs a commitment key, the description of a PPE and a vector of pairs of committed values / randomness. On inputs the commitment key, the equation description, a vector of commitments and a proof, $\mathsf{Verify}(\cdot,\cdot,\cdot)$ outputs a value in $\{0,1\}$. The algorithm $\mathsf{RdProof}(\cdot,\cdot,\cdot)$ takes as inputs a commitment key, an equation description, a vector of pairs of commitments and fresh randomness, and a proof; and outputs a new proof adapted to the randomizations of the commitments.
- Automorphic signatures: let $\mathbf{Sig} = (\mathsf{Setup}_S, \mathsf{KeyGen}_S, \mathsf{Sign}_S, \mathsf{Ver}_S)$ denote Scheme 2 from Sect. 4.1 (p. 37) and $\mathbf{Sig}'$ be the pair transformation of $\mathbf{Sig}$ (Definition 2, p. 41). For $vk = (X, Y)$, $msg = (M, N)$ and $\Sigma = (A, C, D, R, S)$, let $\mathrm{E}_{\mathsf{Ver}(vk,msg,\Sigma)}$ denote the equations in (4.2) and $e(M, H) = e(G, N)$. (We implicitly assume fixed parameters (G, F, K, T, H).) Analogously, let $\mathrm{E}_{\mathsf{Ver}'(vk,(msg_1,msg_2),\Sigma)}$ be the verification relations for a signature of the pair transform on 2 DH-pairs.

We now formally describe the scheme.

$\mathsf{Setup}_{\mathrm{aps}}(1^\lambda)$

- Generate a bilinear group grp for security parameter λ.
- Run $\mathsf{Setup}_{\mathrm{S}}(grp)$ to get parameters pp_{S}.
- Run $\mathsf{KeyGen}_{\mathrm{S}}(pp_{\mathrm{S}})$ to produce a key pair (ipk, ik). Return the public parameters $pp := (pp_{\mathrm{S}}, ipk)$ and the issuer's key ik.

$\mathsf{Reg}_{\mathrm{aps}}$ is a protocol between a new user, the issuer and the user's opener.

- The user runs $(vk, sk) \leftarrow \mathsf{KeyGen}_{\mathrm{S}}(pp_{\mathrm{S}})$ and produces a signature (possibly via an external PKI[4]) Σ_{pki} on vk. She sends $(vk, \Sigma_{\mathrm{pki}})$ to the issuer and vk to the opener.
- The issuer checks Σ_{pki}, produces $cert \leftarrow \mathsf{Sign}_{\mathrm{S}}(ik, vk)$, sends $cert$ to the user, and writes $(vk, \Sigma_{\mathrm{pki}})$ to its register.
- The opener runs $(ck, ek) \leftarrow \mathsf{ExSetup}(grp)$ and sends ck to the user. It sets the opening key as $ok := (vk, ck, ek)$.
- The user sets his public key $upk = (vk, ck)$ and his secret key $usk = (upk, sk, cert)$.

$\mathsf{Delgt}_{\mathrm{aps}}(usk, [\boldsymbol{warr}], upk)$

- Set $k = 0$ if this is an original delegation (i.e., there is no optional argument $\boldsymbol{warr}$), otherwise let k be s.t. this is the k-th intermediate delegation. Parse usk as $\big((vk_k, ck_k), sk_k, cert_k\big)$ and the delegatee's public key upk as (vk_{k+1}, ck_{k+1}).
- If $k = 0$ then choose an identifier id, compute $warr_{0\to1} \leftarrow \mathsf{Sign}'_{\mathrm{S}}\big(sk_0, (\mathcal{H}(id \,\|\, 1), vk_1)\big)$ and return $(ck_0, id, vk_0, warr_{0\to1})$.
- If $k = 1$ then do the following:
  - Parse $\boldsymbol{warr}$ as $(ck, id, vk_0, warr_{0\to1})$.
  - Compute $warr_{1\to2} \leftarrow \mathsf{Sign}'_{\mathrm{S}}\big(sk_1, (\mathcal{H}(id \| 2), vk_2)\big)$.
  - Choose $\xi, \gamma, \omega_1, \omega_2$ and compute the following commitments and proofs:
    $\mathbf{c}_{warr_{0\to1}} \leftarrow \mathsf{Com}(ck, warr_{0\to1}, \omega_1)$, $\mathbf{c}_{vk_1} \leftarrow \mathsf{Com}(ck, vk_1, \xi)$,
    $\mathbf{c}_{cert_1} \leftarrow \mathsf{Com}(ck, cert_1, \gamma)$,
    $\mathbf{c}_{warr_{1\to2}} \leftarrow \mathsf{Com}(ck, warr_{1\to2}, \omega_2)$, $\mathbf{c}_{vk_2} \leftarrow \mathsf{Com}(ck, vk_2, 0)$,
    $\pi_{cert_1} \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathsf{Ver}(ipk,\cdot,\cdot)}, ((vk_1, \xi_1), (cert_1, \gamma))\big)$,
    $\pi_{warr_{0\to1}} \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathsf{Ver}'(vk_0,(\mathcal{H}(id\|1),\cdot),\cdot)}, \big((vk_1, \xi), (warr_{0\to1}, \omega_1)\big)\big)$,
    $\pi_{warr_{1\to2}} \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathsf{Ver}'(\cdot,(\mathcal{H}(id\|2),\cdot),\cdot)}, \big((vk_1, \xi), (vk_2, 0), (warr_{1\to2}, \omega_2)\big)\big)$.
  - Return $\boldsymbol{warr}' := \big(ck, id, vk_0, (\mathbf{c}_{warr_{0\to1}}, \pi_{warr_{0\to1}}, \mathbf{c}_{vk_1}, \mathbf{c}_{cert_1}, \pi_{cert_1}), \mathbf{c}_{warr_{1\to2}}, \pi_{warr_{1\to2}}, \mathbf{c}_{vk_2}\big)$.
- Otherwise, do the following:
  - Parse $\boldsymbol{warr}$ as $\big(ck, id, vk_0, (\mathbf{c}_{warr_{(i-1)\to i}}, \pi_{warr_{(i-1)\to i}}, \mathbf{c}_{vk_i}, \mathbf{c}_{cert_i}, \pi_{cert_i})_{i=1}^{k-1}, \mathbf{c}_{warr_{(k-1)\to k}}, \pi_{warr_{(k-1)\to k}}, \mathbf{c}_{vk_k}\big)$.
  - Compute $warr_{k\to(k+1)} \leftarrow \mathsf{Sign}'_{\mathrm{S}}\big(sk_k, (\mathcal{H}(id \| k+1), vk_{k+1})\big)$.

[4] To achieve strong notions of non-frameability, it is necessary to assume an external public-key infrastructure (PKI) (cf. [BSZ05])

– Choose randomness for commitments and randomization: Pick $\xi_i, \gamma_i, \omega_i$ for $1 \le i \le k$ and ω_{k+1}.

– Randomize the commitments and adapt the proofs in $\boldsymbol{warr}$:
For $1 \le i \le k$: $\mathbf{c}'_{warr_{(i-1)\to i}} \leftarrow \mathsf{RdCom}(ck, \mathbf{c}_{warr_{(i-1)\to i}}, \omega_i)$, $\mathbf{c}'_{vk_i} \leftarrow \mathsf{RdCom}(ck, \mathbf{c}_{vk_i}, \xi_i)$,
$\pi'_{warr_{(i-1)\to i}} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\mathsf{Ver}'(\cdot,(\mathcal{H}(id\|i),\cdot),\cdot)}, ((\mathbf{c}_{vk_{i-1}}, \xi_i{-}1), (\mathbf{c}_{vk_i}, \xi_i), (\mathbf{c}_{warr_{(i-1)\to i}}, \omega_i)), \pi_{warr_{(i-1)\to i}}\big)$.
For $1 \le i \le k-1$: $\mathbf{c}'_{cert_i} \leftarrow \mathsf{RdCom}(ck, \mathbf{c}_{cert_i}, \gamma_i)$,
$\pi'_{cert_i} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\mathsf{Ver}(ipk,\cdot,\cdot)}, \big((\mathbf{c}_{vk_i}, \xi_i), (\mathbf{c}_{cert_i}, \gamma_i)\big), \pi_{cert_i}\big)$.

– Compute the following commitments and proofs:
$\mathbf{c}_{cert_k} \leftarrow \mathsf{Com}(ck, cert_k, \gamma_k)$, $\mathbf{c}_{warr_{k\to(k+1)}} \leftarrow \mathsf{Com}(ck, warr_{k\to(k+1)}, \omega_{k+1})$,
$\mathbf{c}_{vk_{k+1}} \leftarrow \mathsf{Com}(ck, vk_{k+1}, 0)$,
$\pi_{cert_k} \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathsf{Ver}(ipk,\cdot,\cdot)}, ((vk_k, \xi_k), (cert_k, \gamma_k))\big)$
$\pi_{warr_{k\to(k+1)}} \leftarrow \mathsf{Prove}\big(ck, \mathrm{E}_{\mathsf{Ver}'(\cdot,(\mathcal{H}(id\|k+1),\cdot),\cdot)}, ((vk_k, \xi_k), (vk_{k+1}, 0), (warr_{k\to(k+1)}, \omega_{k+1}))\big)$.

– Return $\boldsymbol{warr}' = \big(ck, id, vk_0, (\mathbf{c}'_{warr_{(i-1)\to i}}, \pi'_{warr_{(i-1)\to i}}, \mathbf{c}'_{vk_i}, \mathbf{c}'_{cert_i}, \pi'_{cert_i})_{i=1}^{k-1}, (\mathbf{c}'_{warr_{(k-1)\to k}}, \pi'_{warr_{(k-1)\to k}}, \mathbf{c}'_{vk_k}, \mathbf{c}_{cert_k}, \pi_{cert_k}), \mathbf{c}_{warr_{k\to(k+1)}}, \pi_{warr_{k\to(k+1)}}, \mathbf{c}_{vk_{k+1}}\big)$.

$\mathsf{PSign}_{\mathrm{aps}}(usk, \boldsymbol{warr}, msg)$ Signing is done similarly to delegation, where the message now plays the rôle of vk_{k+1}. Since the message is public, it is not committed to; moreover, ck and vk_0 are part of the verification key and need thus not be included in the signature (see (6.1) below).

$\mathsf{Ver}_{\mathrm{aps}}(upk, msg, \Sigma)$

- Parse upk as (vk_0, ck) and parse the signature Σ as

$$\big(id, (\mathbf{c}_{warr_{(i-1)\to i}}, \pi_{warr_{(i-1)\to i}}, \mathbf{c}_{vk_i}, \mathbf{c}_{cert_i}, \pi_{cert_i})_{i=1}^{k}, \mathbf{c}_{sig}, \pi_{sig}\big) \quad . \tag{6.1}$$

- Return 1 if all of the following return 1, otherwise return 0.

  – $\mathsf{Verify}\big(ck, \mathrm{E}_{\mathsf{Ver}(ipk,\cdot,\cdot)}, (\mathbf{c}_{vk_i}, \mathbf{c}_{cert_i}), \pi_{cert_i}\big)$, for $1 \le i \le k$;
  – $\mathsf{Verify}\big(ck, E_{\mathsf{Ver}'(vk_0,(\mathcal{H}(id\|i),\cdot),\cdot)}, \cdot), (\mathbf{c}_{vk_1}, \mathbf{c}_{warr_{0\to1}}), \pi_{warr_{0\to1}}\big)$;
  – $\mathsf{Verify}\big(ck, E_{\mathsf{Ver}'(\cdot,(\mathcal{H}(id\|i),\cdot),\cdot)}, (\mathbf{c}_{vk_{i-1}}, \mathbf{c}_{vk_i}, \mathbf{c}_{warr_{(i-1)\to i}}), \pi_{warr_{(i-1)\to i}}\big)$, for $2 \le i \le k$;
  – $\mathsf{Verify}\big(ck, E_{\mathsf{Ver}'(\cdot,(\mathcal{H}(id\|k+1),msg),\cdot)}, (\mathbf{c}_{vk_k}, \mathbf{c}_{sig}), \pi_{sig}\big)$.

$\mathsf{Open}_{\mathrm{aps}}(ok, msg, \Sigma)$ Parse ok as (vk, ck, ek), parse Σ as (6.1) and check if it is valid. If so then set $vk_i \leftarrow \mathsf{Extr}(ek, \mathbf{c}_{vk_i})$ and $warr_{(i-1)\to i} \leftarrow \mathsf{Extr}(ek, \mathbf{c}_{warr_{(i-1)\to i}})$ for $1 \le i \le k$, and $sig \leftarrow \mathsf{Extr}(ek, \mathbf{c}_{sig})$. Return $\big((vk_1, \ldots, vk_k), (warr_{0\to1}, \ldots, warr_{(k-1)\to k}, sig)\big)$, where the second component is the proof.

This concludes our description of a proxy signature scheme based on automorphic signatures where the signers and the delegators are anonymous w.r.t. the verifier, and the delegatees are anonymous w.r.t. the delegators.

Chapter 7

Non-interactively Delegatable Anonymous Credentials

Contents

Our main application of commuting signatures and verifiable encryption is a black-box construction of a delegatable anonymous credential scheme with a non-interactive delegation protocol. The scheme borrows the idea of combining Groth-Sahai proofs and automorphic signatures from the instantiation of *anonymous proxy signatures* in the previous chapter. This primitive is similar to delegatable credentials in that it enables to prove knowledge of a certification chain, but there is no mutual anonymity of the users in the delegation protocol. Commuting signatures now allow to define a delegation protocol where both delegator and delegatee remain anonymous w.r.t. each other. Moreover, the protocol is *non-interactive*, that is, a user can publish a pseudonym (a commitment to the user public key), which can be used by the delegator to produce a credential for the user—as it would be in the non-anonymous case with public keys instead of pseudonyms.

We start by presenting the model for delegatable credentials defined in [BCC+09]. In Sect. 7.2 we give our instantiation of it, and compare it to that from [BCC+09] in the subsequent section. The results of this chapter appear in [8].

7.1 The BCCKLS Model

The system parameters are set up by a trusted party. Every user holds a secret key sk, of which she can publish *pseudonyms* Nym. Any user can be the *originator* of a credential by publishing a pseudonym Nym_O as the public key. To issue or delegate a credential, the issuer and the user (both known to each other under their respective pseudonyms) run a protocol at the end of which the user holds a credential. The holder can produce a *credential proof* for a particular pseudonym which proves that the owner of that pseudonym holds a credential rooted at some public key Nym_O.

Thus, if user U was issued a credential under pseudonym Nym_A, she can make a credential proof for any other pseudonym Nym'_A.

In our *non-interactive* instantiation we have the following: To delegate (or to issue) a credential to a user known to the delegator under Nym_U, the delegator produces (without interacting with the user) a ready credential proof for Nym_U. The user can turn this credential proof into a credential, which (as in the BCCKLS model) she can use to make a credential proof for another pseudonym.

A (non-interactively) delegatable anonymous credential system consists of the following algorithms:[1] $\mathsf{Setup}_\mathrm{C}$ generates the parameters, $\mathsf{KeyGen}_\mathrm{C}$ generates secret keys, of which NymGen outputs pseudonyms. Issue produces a credential proof for another user (delegation/issuing), from which that user can obtain an actual credential by running Obtain. With CredProve, a user makes a credential proof for one of his own pseudonyms; and proofs are verified by CredVerify. In detail:

$\mathsf{Setup}_\mathrm{C}(1^\lambda)$ outputs the system parameters pp.

$\mathsf{KeyGen}_\mathrm{C}(pp)$ creates a user secret key sk.

$\mathsf{NymGen}(pp, sk)$ outputs a new pseudonym Nym and auxiliary information aux related to Nym.

$\mathsf{Issue}(pp, Nym_O, sk_I, Nym_I, aux_I, cred, Nym_U, L)$: sk_I, Nym_I and aux_I are the issuer's secret key, pseudonym and auxiliary information. $cred$ is a level L credential for the issuer rooted at Nym_O, and Nym_U is the pseudonym of the delegated user. If $L = 0$ then $cred = \varepsilon$. The algorithm outputs *credproof*.

$\mathsf{Obtain}(pp, Nym_O, sk_U, Nym_U, aux_U, Nym_I, L, credproof)$: sk_U, Nym_U and aux_U are the user's secret key, pseudonym and auxiliary information. Nym_O and Nym_I are the originator's and the issuer's pseudonym, and *credproof* is a proof that the user behind Nym_U holds a level L credential from Nym_O. The algorithm outputs a credential *cred*.

$\mathsf{CredProve}(pp, Nym_O, cred, sk, Nym, aux, L)$ takes a level L credential *cred* from Nym_O for Nym, and sk, Nym and aux, and outputs a *credproof* for Nym.

$\mathsf{CredVerify}(pp, Nym_O, credproof, Nym, L)$ verifies a level L *credproof* for a pseudonym Nym rooted at Nym_O.

Note that CredProve outputs a *credproof* for the user that runs it, while Issue outputs a *credproof* for a different user. Security is defined by *correctness*, *anonymity* and *unforgeability*, which we sketch below. For a formal security definition we refer to Appendix A of the full version of [BCC+09].

Correctness. A credential is *proper* if for all user pseudonyms, CredProve outputs a proof that is accepted by CredVerify. Run honestly, Issue and Obtain must produce a proper credential.

Anonymity. Anonymity means that an adversary interacting with honest users cannot distinguish the real game from an ideal game in which the pseudonyms, credentials and proofs are *independent* of users' identities and credentials. Formally, there exists a simulator ($\mathsf{SimSetup}_\mathrm{C}$, SimCredProve, SimIssue, SimObtain) with the following properties: $\mathsf{SimSetup}_\mathrm{C}$ outputs parameters that are indistinguishable from those produced by Setup and a trapdoor *sim*. Under theses parameters the outputs Nym of NymGen are distributed independently of sk.

SimCredProve gets *sim* instead of $cred, sk$ and aux and outputs *credproof* that is indistinguishable from outputs of CredProve. SimIssue has input *sim* instead of sk_I, aux_I and *cred* and cannot be distinguished from Issue by an adversary interacting with it. SimObtain gets *sim* instead of sk_U and aux_U and cannot be distinguished from Obtain by an adversary interacting with it. All indistinguishability notions hold against adversaries who are given *sim*.

[1] Since, as opposed to [BCC+09], we consider non-interactive delegation, Issue and Obtain are *not interactive* algorithms; the output of Issue is *credproof* which is an additional input to Obtain.

Note that for the case of *non-interactive* delegation, this means: SimIssue produces *credproof* that is indistinguishable from outputs of Issue. And SimObtain is obsolete, since the issuer does not *interact* with it.

Unforgeability. An adversary breaks unforgeability if he produces a proof that some *Nym* has a credential although such a credential has never been issued to any pseudonym of the owner of *Nym*. To formalize the notion of "owner", we define an extraction algorithm that extracts from a pseudonym a user identity *vk*, which is uniquely defined by the secret key. Moreover, from a credential proof it extracts the identities that represent the underlying delegation chain. We say that a forgery occurs if the adversary produces a credential for authority vk_0 from which are extracted $(vk_1, \ldots, vk_{L-1}, vk_L)$ such that vk_{L-1} is the identity of an honest user who never delegated a level L credential rooted at vk_0 to vk_L. Unforgeability is formalized as follows:
(I) There exists $\mathsf{ExSetup}_{\mathrm{C}}$ that outputs parameters *pp* (distributed as those from $\mathsf{Setup}_{\mathrm{C}}$) and an extraction key *ek*. Under *pp* pseudonyms are perfectly binding for *sk* and $\mathsf{Extract}_{\mathrm{C}}$, using *ek*, extracts the identity *vk* from a pseudonym. Given a level L *credproof*, $\mathsf{Extract}_{\mathrm{C}}$ outputs the chain of L identities.
(II) No adversary $\mathcal{A}$ can output a valid credential from which an *unauthorized* chain of identities can be extracted. $\mathcal{A}$ is given the parameters and the extraction key, and has oracles to add honest users, request pseudonyms from them, request issuings between honest users, request proofs and it can run Issue and Obtain with the simulator who plays the role of honest users. When $\mathcal{A}$ requests Issue for $(Nym_O, Nym_I, Nym_U, cred, L)$, the simulator extracts vk_O, vk_I, vk_U from the pseudonyms and adds $(vk_O, L+1, vk_I, vk_U)$ to a list ValidCredentialChains. The adversary wins if it outputs a valid triple $(Nym_O, credproof, Nym)$, from which can be extracted $(vk_0, \ldots, vk_L)$ s.t. $(vk_0, i, vk_{i-1}, vk_i) \notin$ ValidCredentialChains for some i and vk_{i-1} is an honest user identity.

7.2 Our Instantiation

In the instantiation from [BCC+09] the system parameters are a Groth-Sahai (GS) commitment key and parameters for an authentication scheme. Each user holds a secret key x for the authentication scheme. A pseudonym is made up of two GS commitments to H^x and U^x (from which x cannot be extracted), respectively, for parameters H and U. To issue and delegate, the issuer and the user jointly compute a proof of knowledge of an *authenticator* on the user's secret key, which is valid under the issuer's secret key. The authors define a complex interactive two-party protocol for this. A credential is then a chain of pseudonyms and committed authenticators with GS proofs of validity.

We replace the authentication scheme by an automorphic signature scheme. A non-anonymous credential for vk_L rooted at vk_0 is a chain of public keys and signatures $(\Sigma_1, vk_1, \Sigma_2 \ldots, vk_{L-1}, \Sigma_L)$, where Σ_i is a signature on vk_i under vk_{i-1}. To achieve anonymity, the public keys and signatures in the credential are committed to and proofs of validity are added. Using commuting signatures, given a commitment to a public key, the issuer can directly make a commitment to a signature on it and a validity proof. This is what enables non-interactive delegation.

Commuting Signatures with Partially Public Messages. To instantiate credentials, merely signing user public keys does not suffice. The issuer of a credential might want to add public information to the credential, such as attributes. For delegatable credentials it is also required to include the originator's pseudonym and the delegation level in each certificate to prevent combining different credentials and changing the order within a credential.

In Sect. 7.3.1, we give an automorphic signature scheme $\mathbf{Sig}''$, where in addition to the message, the signer can specify some public value. The message space of $\mathbf{Sig}''$ is $\mathbb{Z}_p \times \mathcal{M}$. Our scheme only extends the parameters of pp by one group element, but is otherwise as efficient as **Sig**, in particular, **Sig**- and $\mathbf{Sig}''$ signatures have the same size. For $\mathbf{Sig}''$ we also define an algorithm VK that on input a signing key outputs the corresponding verification key, which allows us to comply with the formal definition of credentials in [BCC+09]. In Sect. 7.3.2, we define SigCom'', which is SigCom adapted to $\mathbf{Sig}''$ and thus has the public part of the message as additional input. Moreover, we show that all the other algorithms defined in Definition 5 and instantiated in Sect. 5.4 work equally for **Sig** and $\mathbf{Sig}''$.

For our instantiation, we assume a collision-resistant hash function $\mathcal{H}\colon \mathcal{C}_{\mathcal{M}} \times \mathbb{N} \to \mathbb{Z}_p$.

More Intuition. We informally describe how our algorithms work. $\mathsf{Setup}_{\mathrm{C}}$ generates a key for **Com** and parameters for $\mathbf{Sig}''$, $\mathsf{KeyGen}_{\mathrm{C}}$ outputs a signing key for $\mathbf{Sig}''$, and given a secret key, NymGen outputs a commitment to the corresponding verification key and the used randomness as auxiliary information. A level L credential proof from Nym_0 to Nym_L has the form

$$credproof = (\mathbf{c}_1, \pi_1, Nym_1, \mathbf{c}_2, \pi_2, \ldots, Nym_{L-1}, \mathbf{c}_L, \pi_L) \ ,$$

where $\mathbf{c}_i$ is a commitment to a signature Σ_i on the public value $\mathcal{H}(Nym_0, i)$ and the key committed in Nym_i, valid under the key committed in Nym_{i-1}; and π_i is a proof of validity of Σ_i. We call it a *credential* if it is valid on a *trivial* Nym_L, i.e., when $Nym_L = \mathsf{Com}(ck, vk_L, 0)$.

$\mathsf{CredProve}$ takes a credential and turns it into a credential proof by randomizing all its components, but using aux s.t. $Nym_L = \mathsf{Com}(ck, \mathsf{VK}(sk), aux)$ for the last component. $\mathsf{CredVerify}$ verifies a *credproof* by checking the proofs contained in it. Given a level L credential, Issue extends it by one level and makes a credential proof for Nym_{L+1}: if it is not an original issuing, it first makes a *credproof* for the issuer's pseudonym Nym_I; using SigCom'' it produces $(\mathbf{c}_{L+1}, \pi_{L+1})$ for Nym_{L+1}, and turns π_{L+1} into a proof for the committed verification key Nym_I by running $\mathsf{AdPrC}_{\mathcal{K}}$ on randomness aux_I. Obtain turns a credential proof into a credential by adapting the randomness to make it valid for a trivial Nym_L.

Algorithm Specification. We now formally define the algorithms of our scheme **Cred**.

$\mathsf{Setup}_{\mathrm{C}}(1^\lambda)$. Run $ck \leftarrow \mathsf{Setup}$; $pp_{\mathrm{S}} \leftarrow \mathsf{Setup}''_{\mathrm{S}}$; return $pp := (ck, pp_{\mathrm{S}})$, which defines $\mathsf{Ver}'', \mathcal{M}, \mathcal{R}_{\mathcal{M}}, \mathcal{C}_{\mathcal{M}}$

$\mathsf{KeyGen}_{\mathrm{C}}(pp)$. Parse $pp \rightsquigarrow (ck, pp_{\mathrm{S}})$; run $(vk, sk) \leftarrow \mathsf{KeyGen}''_{\mathrm{S}}(pp_{\mathrm{S}})$; return sk

$\mathsf{NymGen}(pp, sk)$. Choose $aux \leftarrow \mathcal{R}_{\mathcal{M}}$; return $\big(Nym := \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk), aux),\ aux\big)$

$\mathsf{Issue}(pp, Nym_O, sk_I, Nym_I, aux_I, cred, Nym_U, L)$.

- If $Nym_I \neq \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk_I), aux_I)$ or $Nym_U \notin \mathcal{C}_{\mathcal{M}}$ then abort
- Case $L = 0$: if $cred \neq \varepsilon$ or $Nym_O \neq Nym_I$ then abort
  Case $L > 0$: set $credproof \leftarrow \mathsf{CredProve}(pp, Nym_O, cred, sk_I, Nym_I, aux_I, L)$ and abort if it fails
- $(\mathbf{c}_{L+1}, \pi_{L+1}) \leftarrow \mathsf{SigCom}''\big(pp, sk_I, \mathcal{H}(Nym_O, L+1), Nym_U\big)$
  $\pi'_{L+1} \leftarrow \mathsf{AdPrC}_{\mathcal{K}}\big(pp, (\mathsf{VK}(sk_I), aux_I), Nym_U, \mathbf{c}_{L+1}, \pi_{L+1}\big)$
- Case $L = 0$: return $(\mathbf{c}_1, \pi'_1)$
  Case $L > 0$: return $credproof \,\|\, (Nym_I, \mathbf{c}_{L+1}, \pi'_{L+1})$

$\mathsf{Obtain}(pp, Nym_O, sk_U, Nym_U, aux_U, Nym_I, L;\ credproof)$.

- Case $L = 0$: if $Nym_O \neq Nym_I$ then abort; parse $credproof \rightsquigarrow (\mathbf{c}_1, \pi_1)$
  Case $L > 0$: parse $credproof \rightsquigarrow credproof_L \,\|\, (Nym_{L+1}, \mathbf{c}_{L+1}, \pi_{L+1})$;
  if $Nym_{L+1} \neq Nym_I$ then abort
- If $Nym_U \neq \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk_U), aux_U)$ or $\mathsf{CredVerify}(pp, Nym_O, credproof, Nym_U, L+1) = 0$ then abort
- Parse $pp \rightsquigarrow (ck, pp_{\mathrm{S}})$;
  $\pi'_{L+1} \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\mathsf{Ver}''(\cdot,\mathcal{H}(Nym_O,L+1),\cdot,\cdot)}, (Nym_I, 0), (Nym_U, -aux_U), (\mathbf{c}_{L+1}, 0), \pi_{L+1}\big)$
- Case $L = 0$: return $(\mathbf{c}_1, \pi'_1)$
  Case $L > 0$: return $credproof_L \,\|\, (Nym_{L+1}, \mathbf{c}_{L+1}, \pi'_{L+1})$

$\mathsf{CredProve}(pp, Nym_O, cred, sk, Nym, aux, L)$.

- If $Nym \neq \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk), aux)$ or $\mathsf{CredVerify}\big(pp, Nym_O, cred, \mathsf{Com}_{\mathcal{M}}(pp, \mathsf{VK}(sk), 0), L\big) = 0$ then abort
- Parse $pp \rightsquigarrow (ck, pp_{\mathrm{S}})$ and $cred \rightsquigarrow (\mathbf{c}_1, \pi_1, Nym_1, \mathbf{c}_2, \pi_2, \ldots, Nym_{L-1}, \mathbf{c}_L, \pi_L)$
- For $i = 1 \ldots L$, pick $\nu_i \leftarrow \mathcal{R}_{\mathcal{M}}$, $\gamma_i \leftarrow \mathcal{R}$. Set $Nym_0 := Nym_O$, $\nu_0 := 0$, $Nym_L := Nym$, $\nu_L := aux$
- For $i = 1 \ldots L$ do
  $Nym'_i := \mathsf{RdCom}_{\mathcal{M}}(pp, Nym_i, \nu_i)$; $\mathbf{c}'_i := \mathsf{RdCom}(ck, \mathbf{c}_i, \gamma_i)$
  $\pi'_i \leftarrow \mathsf{RdProof}\big(ck, \mathrm{E}_{\mathsf{Ver}''(\cdot,\mathcal{H}(Nym_O,i),\cdot,\cdot)}, (Nym_{i-1}, \nu_{i-1}), (Nym_i, \nu_i), (\mathbf{c}_i, \gamma_i), \pi_i\big)$
- Return $(\mathbf{c}'_1, \pi'_1, Nym'_1, \mathbf{c}'_2, \pi'_2, \ldots, Nym'_{L-1}, \mathbf{c}'_L, \pi'_L)$

$\mathsf{CredVerify}(pp, Nym_O, credproof, Nym, L)$

- Parse $pp \rightsquigarrow (ck, pp_{\mathrm{S}})$, $credproof \rightsquigarrow (\mathbf{c}_1, \pi_1, Nym_1, \ldots, \mathbf{c}_L, \pi_L)$,
  let $Nym_0 := Nym_O$, $Nym_L := Nym$
- If $\forall\ 1 \leq i \leq L : \mathsf{Verify}\big(ck, \mathrm{E}_{\mathsf{Ver}''(\cdot,\mathcal{H}(Nym_O,i),\cdot,\cdot)}, Nym_{i-1}, Nym_i, \mathbf{c}_i, \pi_i\big) = 1$ and $Nym_i \in \mathcal{C}_{\mathcal{M}}$, then return 1

Theorem 10. *Let* $(\mathbf{Com}, \mathbf{Proof}, \mathbf{Sig}'', \mathbf{Com}_{\mathcal{M}}, \mathbf{Algs})$ *be a commuting signature system where* $(\mathbf{Com}, \mathbf{Proof})$ *is randomizable, extractable and composable zero-knowledge*[2] *and* $\mathbf{Sig}''$ *is automorphic. Let moreover* $\mathcal{H}$ *be a collision resistant hash function. Then* **Cred** *as defined above is a secure delegatable anonymous credential scheme.*

Instantiating the commuting signature as described in Sect. 7.3, the resulting credential scheme is secure under ADH-SDH and SXDH.

Proof sketch. We refer to the full version of [BCC+09] for the formal definition of the model, which is quite involved. Since the overall construction of our scheme is similar to the BCCKLS construction, in particular the use of (randomizable and simulatable) Groth-Sahai proofs to commit to a delegation chain and prove validity, our scheme is proved to satisfy the security definitions analogously. The proof is a lot simpler though, since our certificates are on *public keys* and one can extract a complete certification chain from our credentials, avoiding thus partial-extractability notions. Moreover, our construction does not make use of interactive secure two-party protocols. We give a sketch of the security proof, highlighting the differences.

[2] A proof system is *composable* zero-knowledge if for a simulated CRS simulated proofs are indistinguishable from regular proofs even if the distinguisher is given the simulation trapdoor.

Correctness. Correctness of our scheme follows by a straightforward argument from correctness and completeness of the underlying building blocks.

Anonymity. A witness-indistinguishability based definition of anonymity is an immediate consequence of perfectly hiding commitments and proofs, when *ck* is produced by WISetup: pseudonyms *Nym* then information-theoretically hide the committed value and the proofs in *credproof* do not contain information either. We thus define $\mathsf{SimSetup}_{\mathrm{C}}$ using WISetup. The algorithms CredProve, Issue and Obtain can be simulated without knowledge of any private information; this follows from the zero-knowledge property of Groth-Sahai proofs; in particular, in the witness-indistinguishable (WI) setting, given the simulation trapdoor *sim* of GS proofs, we can make perfectly hiding commitments and proofs for any equation with certain properties—which are satisfied by ours. (See Sect. 7.4.)

Our simulator SimCredProve is the exact analogue of SimProve, defined in the anonymity proof of the BCCKLS scheme: it constructs a simulated certificate chain from Nym_O to Nym_I of length L, by simulating the intermediate pseudonyms, i.e., the $\mathbf{Com}_{\mathcal{M}}$ commitments (cf. Sect. 7.4.2), the commitments in $\mathbf{c}_i$, and the proofs π_i. SimIssue is defined as SimCredProve for $L+1$ except that it sets Nym_L to be Nym_I. Since Obtain does not interact with the issuer, SimObtain is the empty algorithm. Our proof is considerably simpler than that of [BCC+09], due to the fact that Issue readily outputs a credential rather than engaging in a two-party protocol with Obtain. All we need to do is simulate GS commitments and proofs. In Sect. 7.4 we discuss how to simulate the proofs output by SimCredProve and SimIssue. These algorithms must simulate proofs for equations for which some of the commitments (Nym_O and Nym_I) are given, which was not considered in the simulations defined in [GS08].

Unforgeability. Soundness and extractability of GS proofs, unforgeability of $\mathbf{Sig}''$ and simulatability of SigCom'' imply that our scheme is unforgeable in the sense of [BCC+09]: (I) We define $\mathsf{ExSetup}_{\mathrm{C}}$ as $\mathsf{Setup}_{\mathrm{C}}$ in which we substitute Setup by ExSetup. This generates an identically distributed key *ck* (which leads to perfectly binding commitments) and an extraction key *ek* that allows to extract the committed chain of verification keys ("identities") and certificates from a credential.

The notion defined in (II) is reduced to unforgeability of $\mathbf{Sig}''$: given a verification key *vk* and a signing oracle, we simulate the game as follows: we guess which honest user the adversary will "frame"; we compute the parameters with $\mathsf{ExSetup}_{\mathrm{C}}$, and use extraction, the signing oracle and SmSigCom to simulate Issue for that user. (This is done analogously to the reduction for unforgeability in Sect. 5.2.1, p. 50.) Let $(Nym_0, \mathit{credproof}, Nym_L)$ be a successful forgery, thus when we extract $(vk_0, \Sigma_1, vk_1, \ldots, \Sigma_L, vk_L)$ from it then for some i: $(vk_0, i, vk_{i-1}, vk_i) \notin \mathsf{ValidCredentialChains}$ and vk_{i-1} is honest. If we guessed correctly (i.e., $vk_{i-1} = vk$) then we can return the $\mathbf{Sig}''$ forgery (vk_i, Σ_i), since (by collision resistance of $\mathcal{H}$) we have never queried our signing oracle on $(\mathcal{H}(Nym, i), vk_i)$ for any *Nym* of vk_0. □

Optimizing the Black-Box Construction for Concrete Commuting Signatures. When using our implementation of commuting signatures (Sect. 5.4), we can make the following optimizations. In the instantiation we have $\mathcal{M} \subseteq \mathbb{G}_1 \times \mathbb{G}_2$ for an asymmetric bilinear group. A $\mathbf{Com}_{\mathcal{M}}$ commitment to a message $(M, N) \in \mathcal{M}$ is defined as $\mathbf{c}_M := \mathsf{Com}(ck, M, \mu)$, $\mathbf{c}_N := \mathsf{Com}(ck, N, \nu)$, $\pi_N \leftarrow \mathsf{Prove}(ck, \mathrm{E}_{\mathcal{DH}}, (M, \mu), (N, \nu))$ and additional components which enable the signer to make a committed signature on (M, N) and a proof of validity by running SigCom. These additional components are however not required for the *Nym*'s contained in the credential, where giving $(\mathbf{c}_M, \mathbf{c}_N, \pi_M)$ is sufficient. Analogously, the "public keys" Nym_O at which a credential is rooted can also be given in the reduced form.

7.2.1 A Comparison to the BCCKLS Instantiation

The Certification Scheme. The key building block of a delegatable credential scheme is a certification scheme signing (or authenticating) user keys and some public information. The authenticators on user secret keys in the BCCKLS instantiation [BCC+09] are in $\mathbb{G}_1^8 \times \mathbb{G}_2^3$ and are verified by evaluating 16 pairings. Our certificates on a user verification key (and a public value) are in $\mathbb{G}_1^3 \times \mathbb{G}_2^2$ and verified by evaluating 7 pairings. Proving validity of a committed certificate requires Groth-Sahai proofs (of which one is in $\mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$ for the SXDH instantiation) for 8 equations for the BCCKLS instantiation and 3 equations for ours. The pseudonyms in [BCC+09] consist of two Groth-Sahai commitments and one proof, and are thus in $\mathbb{G}_1^6 \times \mathbb{G}_2^6$, as are the pseudonyms contained in our credentials. The pseudonyms enabling non-interactive delegation are the size of two optimized pseudonyms and 3 $\mathbb{G}_1$ elements.[3] If we use WI Groth-Sahai proofs (without considering simulatability), we have the following: a tuple $(\mathbf{c}_i, \pi_i, Nym_i)$ representing one level in a credential is in $\mathbb{G}_1^{54} \times \mathbb{G}_2^{44}$ for the BCCKLS instantiation, whereas it is in $\mathbb{G}_1^{24} \times \mathbb{G}_2^{22}$ for ours. We now analyze how the actual simulatable schemes are instantiated

Simulating Proofs. Proofs for homogeneous equations (i.e., with right-hand side 1_T) can be simulated directly. To make the type of *inhomogeneous* equations used in [BCC+09] and our instantiation simulatable, a Groth-Sahai proof is augmented by one commitment in $\mathbb{G}_1^2$, one commitment in $\mathbb{G}_2^2$ and one proof from $\mathbb{G}_1^4 \times \mathbb{G}_2^2$ (see Sect. 7.4.2, p. 87).

Groth and Sahai only consider simulation of a proof of satisfiability of equations; the simulator can thus make the commitments himself. In contrast, to simulate Issue and CredProve, we have to simulate proofs for which some commitments, like *Nym* are given to the simulator. To make a proof of knowledge of an authenticator, Belenkiy et al. add extra commitments to the message and to the key and a simulatable proof of equality of the committed values. After some optimization the cost for such a commitment and a proof is $\mathbb{G}_1^8 \times \mathbb{G}_2^8$.

In our construction we employ a different strategy. In Sect. 7.4.1 we show that for certain types of equations Groth-Sahai proofs can be simulated *even if some of the commitments are fixed before*, without any additional cost. We then show that all our equations are of this type, which makes the extra commitments and proofs obsolete. Of our 3 equations only 1 is inhomogeneous. We show that the additional commitment and proof needed for simulatability can be used for *all levels* of the credential. A level L credential proof $(\mathbf{c}_1, \pi_1, \ldots, Nym_{L-1}, \mathbf{c}_L, \pi_L)$ consists thus of $24\,L$ elements from $\mathbb{G}_1$ and $22(L-1)+20$ elements from $\mathbb{G}_2$. The extra commitments and proofs in the BCCKLS instantiation have to be added at each level; a credential proof is thus in $\mathbb{G}_1^{62(L-1)+80} \times \mathbb{G}_2^{52(L-1)+54}$. We conclude that the size of our credentials is less than half the size of BCCKLS credentials.

Delegation. Most importantly, issuing and delegation in our scheme are substantially more efficient than in the BCCKLS scheme. In the latter, besides a credential proof for his pseudonym Nym_I, the issuer first sends a GS proof of knowledge of the first 6 components of the authenticator, which do not depend on the message, i.e., the user secret key. The issuer and the user then run a two-party protocol to jointly compute the last component, using a homomorphic encryption scheme and interactive ZK proofs that blinding values are in the correct ranges.

The authors suggest using Paillier encryption [Pai99] based on an RSA modulus of size at least $2^{3k}p^2$. Using the NIST recommendations from 2007 [NIS07] for $k = 128$ bits of security, the RSA modulus N must be at least 2^{3072}; Paillier ciphertexts are thus of size $N^2 \geq 2^{6144}$. Since the interactive proofs of knowledge of plaintexts and values lying in certain intervals are not given

[3] Note that if the size of pseudonyms is to be minimized, users could publish $Nym = (\mathbf{c}_M, \mathbf{c}_N, \pi_M)$ and send the remaining elements $(\mathbf{c}_P, \mathbf{c}_Q, \pi_P, U, \pi_{U_S})$ as a first step in Obtain. See Sect. 7.4.2 for why π_{U_S} is in $\mathbb{G}_1^2$.

explicitly, it is not clear how many rounds of interaction the protocol requires and how many elements are sent in each of them.

In our issuing protocol this part corresponds to simply running SigCom and $\mathsf{AdPrC}_{\mathcal{K}}$ and sending (in addition to the *credproof*) the triple $(Nym_I, \mathbf{c}_{L+1}, \pi'_{L+1})$, which is in $\mathbb{G}_1^{24} \times \mathbb{G}_2^{22}$. For 128-bit security, using e.g. the groups suggested by Barreto and Naehrig [BN05], elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ are represented by 256 and 512 bits, respectively. The size of a triple above is then less than 3 Paillier ciphertexts for comparable security parameters.

Security. Concerning the assumptions on which security is based, they are both *non*-interactive, "q-type" assumptions and part of the generalized "Uber-Assumption" family [Boy08]. What is more, both are comparable variants of the *strong Diffie-Hellman* assumption [BB04], as we argued in Sect. 3.3.1 (p. 31).

7.3 Commuting Signatures with Partially Public Messages

7.3.1 Automorphic Signatures on an Integer and a Message

The scheme **Sig** from Sect. 4.1 can be adapted to sign a value from $\mathbb{Z}_p$ and an element from $\mathcal{DH}$ at the same time, as it is required for our application to delegatable credentials. Note that while this requires one extra element in the parameters it does *not* increase the size of a signature.

Intuition. ADH-SDH states that given a public key (G^x, H^x) and "weak signatures" $((K \cdot V)^{\frac{1}{x+c}}, F^c, H^c)$ on random messages $(V, W) \in \mathcal{DH}$, it is hard to forge such a signature on a new message. Now to turn this into a CMA secure scheme (Scheme 2, p. 37), we implicitly define a *trapdoor commitment* $\mathsf{TCom}((M, N), r) := M \cdot T^r$ with opening (G^r, H^r). The actual signature is then a weak signature on $\mathsf{TCom}((M, N), r)$ together with the opening (G^r, H^r). AWF-CDH implies that it is hard to open a TCom commitment in two different ways, thus TCom is computationally binding.

In order to sign a message *pair* consisting of an integer value v and a DH-pair (M, N), we replace TCom by TCom'' having an additional parameter L. We define $\mathsf{TCom}''(v, (M, N), r) := L^v \cdot M \cdot T^r$ and the opening as (G^r, H^r), as for TCom. TCom'' is also computationally binding by AWF-CDH: Consider an adversary producing $(v, (M, N), (G^r, H^r))$ and $(v', (M', N'), (G^{r'}, H^{r'}))$ with $(v, M, N) \neq (v', M', N')$ and $\mathsf{TCom}''(v, (M, N), r) = \mathsf{TCom}''(v', (M', N'), r')$; then the case $r \neq r'$ is reducible to AWF-CDH—as for TCom—and $r = r'$ is reducible to CDH, which is implied by AWF-CDH (see the proof of Theorem 11 below). Replacing TCom by TCom'' in **Sig** we get the following.

Scheme 4 (Sig″). $\mathsf{Setup}''_{\mathrm{S}}$ *has input* $grp = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, G, H)$ *and outputs* grp *and additional generators* $F, K, L, T \leftarrow \mathbb{G}_1$. *The message space is* $\mathcal{DH} := \{(G^m, H^m) \mid m \in \mathbb{Z}_p\}$.

$\mathsf{KeyGen}''_{\mathrm{S}}$ *chooses* $x \leftarrow \mathbb{Z}_p$ *and outputs* $(vk = \mathsf{VK}(x),\ sk = x)$, *with* $\mathsf{VK}(x) := (G^x, H^x)$

Sign'' *has inputs a secret key* x *and a message* $(v, (M, N)) \in \mathbb{Z}_p \times \mathcal{DH}$. *It chooses random* $c, r \leftarrow \mathbb{Z}_p$ *and outputs*

$$\left(A := (K \cdot L^v \cdot M \cdot T^r)^{\frac{1}{x+c}},\ B := F^c,\ D := H^c,\ R := G^r,\ S := H^r\right)$$

Ver'' *on inputs a public key* $(X, Y) \in \mathcal{DH}$*, a message* $(v, (M, N)) \in \mathbb{Z}_p \times \mathcal{DH}$ *and a signature* (A, B, D, R, S) *outputs 1 (and 0 otherwise) iff the following hold:*

$$e(A, Y \cdot D) = e(K \cdot L^v \cdot M, H)\, e(T, S) \qquad e(B, H) = e(F, D) \qquad e(R, H) = e(G, S) \tag{7.1}$$

Theorem 11. *Assuming q-ADH-SDH and AWF-CDH,* $\mathbf{Sig}''$ *is strongly existentially unforgeable against adversaries making at most* $q - 1$ *adaptive chosen-message queries.*

Proof. Consider an adversary that after receiving parameters (G, F, K, L, T, H) and public key (X, Y) is allowed to ask for $q-1$ signatures $(A_i, B_i, D_i, R_i, S_i)$ on messages $(u_i, (M_i, N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ of its choice and then outputs $(u, (M, N)) \in \mathbb{Z}_p \times \mathcal{DH}$ and a valid signature (A, B, D, R, S) on it, such that either $(u, (M, N))$ was never queried, or $(u, (M, N)) = (u_i, (M_i, N_i))$ and $(A, B, D, R, S) \neq (A_i, B_i, D_i, R_i, S_i)$. We distinguish three kinds of forgers: An adversary is called of Type I if its output satisfies the following:

$$\forall\, 1 \leq i \leq q-1 : \left[e(T, S \cdot S_i^{-1}) \neq e(L^{u_i} \cdot M_i \cdot L^{-u} \cdot M^{-1}, H) \;\vee\; B \neq B_i\right] \tag{7.2}$$

An adversary is called of Type IIa if its output satisfies

$$\exists\, 1 \leq i \leq q-1 : \left[e(T, S \cdot S_i^{-1}) = e(L^{u_i} \cdot M_i \cdot L^{-u} \cdot M^{-1}, H) \;\wedge\; B = B_i \;\wedge\; S \neq S_i\right] \tag{7.3}$$

otherwise it is called of Type IIb. We will use the first type to break q-ADH-SDH, Type IIa to break AWF-CDH and Type IIb to break CDH, which is implied by AWF-CDH.

Type I Let $\left(G, F, K, X, H, Y, (A_i, B_i, V_i, D_i, W_i)_{i=1}^{q-1}\right)$ be a q-ADH-SDH challenge. It satisfies thus

$$e(A_i, Y \cdot D_i) = e(K \cdot V_i, H) \qquad e(B_i, H) = e(F, D_i) \qquad e(V_i, H) = e(G, W_i)$$

Let $\mathcal{A}$ be a forger of Type I. Choose $t, l \leftarrow \mathbb{Z}_p$ and give parameters $(G, F, K, L := G^l, T := G^t, H)$ and the public key (X, Y) to $\mathcal{A}$. The i-th signing query for $(u, (M_i, N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ is answered as

$$\left(A_i, B_i, D_i, R_i := (V_i \cdot G^{-l \cdot u_i} \cdot M_i^{-1})^{\frac{1}{t}}, S_i = (W_i \cdot H^{-l \cdot u_i} \cdot N_i^{-1})^{\frac{1}{t}}\right).$$

It is easily verified that it satisfies (7.1); and it is correctly distributed since $v_i = \log_G V_i$ is uniformly random in the ADH-SDH instance. If the adverseray produces a valid pair $\left((A, B, D, R, S), (u, (M, N))\right)$ then by the last 2 equations of (7.1), there exist c, r s.t. $B = F^c, D = H^c, R = G^r, S = H^r$, and

$$e(A, Y \cdot D) = e(K \cdot L^u \cdot M, H)\, e(T, S)\ . \tag{7.4}$$

The tuple $(A, B, D, V := G^{l \cdot u} \cdot M \cdot R^t, W := H^{l \cdot u} \cdot N \cdot S^t)$ satisfies (3.3) on p. 31, since (B, D) and (V, W) are Diffie-Hellman pairs and $e(K \cdot V, H) = e(K \cdot L^u \cdot M \cdot (G^r)^t, H) = e(K \cdot L^u \cdot M, H)\, e(T, S) \overset{(7.4)}{=} e(A, Y \cdot D)$. Moreover, it is a solution for the ADH-SDH instance, since it is a *new* tuple: assume that for some i we have $B = B_i$ and $W = W_i$, that is $H^{l \cdot u} \cdot N \cdot S^t = H^{l \cdot u_i} \cdot N_i \cdot S_i^t$. Since $(M, N), (M_i, N_i) \in \mathcal{DH}$, we have $e(T, S)\, e(L^u \cdot M, H) = e(T, S)\, e(G, H^{l \cdot u} \cdot N) = e(G, H^{l \cdot u} \cdot N \cdot S^t) = e(G, H^{l \cdot u_i} \cdot N_i \cdot S_i^t) = e(T, S_i)\, e(G, H^{l \cdot u_i} \cdot N_i) = e(T, S_i)\, e(L^{u_i} \cdot M_i, H)$. We have thus $e(T, S \cdot S_i^{-1}) = e(L^{u_i} \cdot M_i \cdot L^{-u} \cdot M^{-1}, H)$ and $B = B_i$ which contradicts (7.2) and thus the fact that $\mathcal{A}$ is of Type I.

Type IIa Let $(G, H, T = G^t)$ be an AWF-CDH instance; let $\mathcal{A}$ be a forger of Type IIa. Pick $F, K \leftarrow \mathbb{G}_1$ and $l, x \leftarrow \mathbb{Z}_p$, set $X := G^x$, $Y := H^x$ and give the adversary parameters $(G, F, K, L := G^l, T, H)$ and public key (X, Y). Answer a signing query on $(u_i, (M_i, N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ by returning a signature $(A_i, B_i, D_i, R_i, S_i)$ produced by $\mathsf{Sign}(x, \cdot)$. Suppose $\mathcal{A}$ returns $\big((A, B, D, R, S), (u, (M, N))\big)$ satisfying (7.1) s.t. $e(T, S \cdot S_i^{-1}) = e(L^{u_i} \cdot M_i \cdot L^{-u} \cdot M^{-1}, H)$, $B = B_i$ and $S \neq S_i$ for some i. Then $(M^* := L^{u_i} \cdot M_i \cdot L^{-u} \cdot M^{-1}, N^* := H^{l \cdot u_i} \cdot N_i \cdot H^{-l \cdot u} \cdot N^{-1}, R^* := R \cdot R_i^{-1}, S^* := S \cdot S_i^{-1})$ is a AWF-CDH solution: (S^*, M^*), (M^*, N^*) and (R^*, S^*) satisfy the respective equations in (3.1), and since $S \neq S_i$ it is non-trivial.

Type IIb Let $(G, H, L := G^l)$ be a CDH instance, i.e., we have to produce H^l. Let $\mathcal{A}$ be a forger of Type IIb. Pick $F, K, T \leftarrow \mathbb{G}_1$ and $x \leftarrow \mathbb{Z}_p$, set $X := G^x$, $Y := H^x$ and give the adversary parameters (G, F, K, L, T, H) and public key (X, Y). Answer a signing query on $(u_i, (M_i, N_i)) \in \mathbb{Z}_p \times \mathcal{DH}$ by returning a signature $(A_i, B_i, D_i, R_i, S_i)$ produced by $\mathsf{Sign}(x, \cdot)$. Suppose $\mathcal{A}$ returns $\big((A, B, D, R, S), (u, (M, N))\big)$ satisfying (7.1) of Type IIa, i.e., $e(T, S \cdot S_i^{-1})$ $= e(L^{u_i} \cdot M_i \cdot L^{-u} \cdot M^{-1}, H)$, $B = B_i$ and $S = S_i$ for some i; which implies $L^{u_i} \cdot M_i = L^u \cdot M$. We first show that $u \neq u_i$: Suppose $u = u_i$; then by the above we have $M = M_i$, and moreover $B = B_i$ and $S = S_i$. Since these values completely determine A, D, R and N, we have $(A, B, D, R, S, u, M, N) = (A_i, B_i, D_i, R_i, S_i, u_i, M_i, N_i)$, which means that $\mathcal{A}$ did not break strong unforgeability.

From $L^{u_i} \cdot M_i = L^u M$ we have $L^{u-u_i} = M_i \cdot M^{-1}$ and since $u \neq u_i$ we have $L = (M_i \cdot M^{-1})^{\frac{1}{u-u_i}}$, which for $m := \log_G M = \log_H N$, $m_i := \log_G M_i = \log_H N_i$ can be written as $G^{\frac{m_i - m}{u-u_i}}$. Thus $(N_i \cdot N^{-1})^{\frac{1}{u-u_i}} = H^{\frac{m_i-m}{u-u_i}}$ is a CDH solution.

□

7.3.2 Verifiably Encrypting a Signature on a Public Integer and a Committed Message

A commitment to a signature on an integer v and a message committed in $\mathbf{C}$ is of the form $(\mathbf{c}_A, \mathbf{c}_B, \mathbf{c}_D, \mathbf{c}_R, \mathbf{c}_S)$ and a proof of validity is $(\pi_{A''}, \pi_B, \pi_R)$ for equations E_B and E_R as in (5.5) on p. 56 and $\mathrm{E}_{A''}$ defined as

$$\mathrm{E}_{A''}(A, M;\ S, D)\ :\ e(T^{-1}, \underline{S})\, e(\underline{A}, Y) e(\underline{M}, H^{-1})\, e(\underline{A}, \underline{D}) = e(K \cdot L^v, H)\ . \tag{7.5}$$

Note that the left-hand sides of $\mathrm{E}_{A''}$ and E_A from (5.5) are equal. By Lemma 3 (p. 51), proofs are independent of the right-hand side of the equation, thus $\pi_{A''}$ is defined like π_A. This also holds for proofs about all other equations such as $\mathrm{E}_{\widetilde{A}}, \mathrm{E}_{\bar{A}}$ and $\mathrm{E}_{\widehat{A}}$ and their variants for $\mathbf{Sig}''$, since going from $\mathbf{Sig}$ to $\mathbf{Sig}''$ only affects the right-hand sides of the equations.

Proofs for $\mathrm{E}_{\mathsf{Ver}(\cdots)}$ and $\mathrm{E}_{\mathsf{Ver}''(\cdots)}$ are thus the same for all combinations of keys, messages and signatures given as commitments or in the clear. This means that the proof-adaptation algorithms $\mathsf{AdPrC}, \mathsf{AdPrC}_{\mathcal{M}}, \mathsf{AdPrC}_{\mathcal{K}}, \mathsf{AdPrDC}, \mathsf{AdPrDC}_{\mathcal{M}}$ and $\mathsf{AdPrDC}_{\mathcal{K}}$ defined in Sect. 5.4.3 can all be used for proofs about committed Sig'' signatures as well. The only functionality that has to be slightly adapted is SigCom. We define $\mathsf{SigCom}''(ck, sk, v, \mathbf{C})$ as SigCom in Figure 5.1, except that A is defined as $A := (K \cdot L^v \cdot T^r \cdot U)$. We do not need to modify $\mathsf{SmSigCom}$, since Σ is given as input to it.

Note that if in the construction of a blind signature in Sect. 5.2.1 we replace $\mathbf{Sig}$ and SigCom by $\mathbf{Sig}''$ and SigCom'', respectively, we obtain *partially blind signatures* [AF96], where the signer controls part of the message.

7.4 Simulating Proofs for Fixed Commitments

Groth and Sahai [GS08] show that pairing-product equations with a right-hand side $\mathfrak{t}_T$ of the form

$$\mathfrak{t}_T \;=\; e(P_1, Q_1) \cdots e(P_n, Q_n) \tag{7.6}$$

can be simulated: in the witness-indistinguishability setting (i.e., when $ck^* \leftarrow \mathsf{WISetup}$; cf. Sect. 2.4.2), given as simulation trapdoor *sim* the values $(\alpha_1, t_1, \alpha_2, t_2)$ used to construct ck^* one can construct commitments and proofs of validity for an equation of the above form *without knowing a witness*, i.e., elements that satisfy the equation.

Equations with right-hand side 1_T ("*homogeneous equations*") can be simulated directly, since they have a trivial witness. Equations with a non-trivial right-hand side as in (7.6) must be transformed to a new set of equations to be simulatable: in the original equation the values P_i are replaced by *variables* V_i (which makes the equation homogeneous) and for each i we add the *multi-scalar multiplication equation*[4] $V_i^d \cdot P_i^{-d} = 1$, where the commitment for d will be a *trivial* commitment to 1 (Since the randomness for the commitment of d is 0, we can check that the committed value is 1, which gives us $V_i = P_i$ from the additional equations, and thus soundness of the construction.) In the simulation, we can now set all variables from $\mathbb{G}_1$ and $\mathbb{G}_2$ to 1 (which is a satisfying witness for our transformed (homogeneous) PPE), and can thus give commitments and proofs. The additional equations can be simulated, since given the trapdoor *sim*, the commitment to d can be trapdoor-opened to 0 (see [GS08] for the details).

In Sect. 7.4.2 we show that modifying our verification equations for commuting signatures does not interfere with its functionality; thus we get simulatability, as required for anonymity of our credentials.

7.4.1 Simulating Proofs of Knowledge with Given Commitments.

Groth and Sahai show that given *sim* in the WI setting, for any simulatable PPE, a simulator can produce commitments and a proof of validity. However, they do not consider the case where some of the commitments are given to the simulator, which means the simulator cannot produce them itself and in particular, it does not know their randomness.

To satisfy the simulation-based anonymity notions for delegatable credentials, simulations of this kind are required: for example, $\mathsf{SimCredProve}$ must produce a credential proof from a given Nym_O for a given *Nym* without being given a credential nor *sk* and *aux* for *Nym*. In the construction of [BCC+09], simulation for fixed commitments is solved in the following way. For every commitment that could be fixed before simulating a proof (i.e., the keys and messages for an authenticator), another commitment is added together with a proof of equality of the committed values. These proofs are then all that have to be simulated for a fixed commitment.

We choose a more direct (and efficient) approach. We show that the proofs used in our construction can all be simulated even when some of the commitments are fixed in advance.

Lemma 8. *Let* E *be as in* (2.2) *on p. 21 with* $\mathfrak{t}_T = 1_T$ *and* $A_j = 1$ *for indices* $j \in J \subseteq \{1, \ldots, n\}$. *Given commitments* $\mathbf{d}_j$ *for* $j \in J$, *we can simulate* $\mathbf{c}_1, \ldots, \mathbf{c}_m$ *and* $\mathbf{d}_j$ *for* $j \notin J$ *and a proof* π *for* E *and* $(\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathbf{d}_1, \ldots, \mathbf{d}_m)$ *if we are given the simulation trapdoor sim for* ck^*. *A symmetric result holds for* $\mathbf{c}_i$ *and* $\mathbf{d}_j$ *interchanged, and* A_j *replaced with* B_i.

[4] An equation of the form $\mathrm{E}(X_1, \ldots, X_m; y_1, \ldots, y_n) : \prod_{i=1}^{n} A_i^{y_i} \prod_{i=1}^{m} X_i^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n} X_i^{\gamma_{ij} \cdot y_j} = T$, over $X_i \in \mathbb{G}_1$ and $y_j \in \mathbb{Z}_p$, is called multi-scalar-multiplication equation in $\mathbb{G}_1$. [GS08] show how to construct WI proofs for this type of equation.

Proof. If the simulator can choose all the commitments $\mathbf{c}_i$ and $\mathbf{d}_j$, it sets the committed values to 1. Since these values satisfy a homogeneous equation, the simulator can make an honest proof using the randomness for the commitments. But if the commitments $(\mathbf{d}_j)_{j\in J}$ are fixed and given to the simulator, it does not know the randomness s_j s.t. $\mathbf{d}_j = \mathsf{Com}(ck^*, 1, s_j)$ for all $j \in J$. We show that the proof can nontheless be construct. Let us look at the definition of $\mathsf{Prove}\big(ck, \mathrm{E}, (X_i, r_i)_{i=1}^m, (Y_j, s_j)_{j=1}^n;\, Z\big)$ in (2.5) on p. 21. For the case when $X_i = 1 = Y_j$ we have

$$t_{11} := \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij}s_{j1} \qquad t_{12} := \sum_{j=1}^n \sum_{i=1}^m r_{i1}\gamma_{ij}s_{j2}$$
$$t_{21} := \textstyle\sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij}s_{j1} \qquad t_{22} := \sum_{j=1}^n \sum_{i=1}^m r_{i2}\gamma_{ij}s_{j2}$$

$$\phi := \begin{bmatrix} v_{11}^{t_{11}} v_{21}^{t_{12}} & \big(\prod_{i=1}^m B_i^{r_{i1}}\big) v_{12}^{t_{11}} v_{22}^{t_{12}} \\ v_{11}^{t_{21}} v_{21}^{t_{22}} & \big(\prod_{i=1}^m B_i^{r_{i2}}\big) v_{12}^{t_{21}} v_{22}^{t_{22}} \end{bmatrix} \circ (Z \otimes \vec{\mathbf{v}}) \qquad \theta := \begin{bmatrix} 1 & \big(\prod_{j=1}^n A_j^{s_{j1}}\big) \\ 1 & \big(\prod_{j=1}^n A_j^{s_{j2}}\big) \end{bmatrix} \circ (Z \otimes \vec{\mathbf{u}})$$

which has to be constructed without knowledge of $(s_j)_{j\in J}$, that is, the values satisfying $\mathbf{d}_j = \mathsf{Com}(ck^*, 1, s_j)$. Let the simulation trapdoor $sim = (\alpha_1, \alpha_2, \beta_1, \beta_2)$ be s.t. $\mathbf{v}_1 = (G_2, G_2^{\alpha_2})$ and $\mathbf{v}_2 = (G_2^{\beta_2}, G_2^{\alpha_2\beta_2 - 1})$ (see Sect. 2.4.2, p. 20). Let (k_j, l_j) be the (unknown) logarithms of $\mathbf{d}_j$, i.e., $d_{j,1} = G_2^{k_j}$ and $d_{j,2} = G_2^{l_j}$. Then we have

$$(G_2^{k_j}, G_2^{l_j}) = \mathbf{d}_j = \mathsf{Com}(ck^*, 1, s_j) = (v_{11}^{s_{j1}} v_{21}^{s_{j2}}, v_{12}^{s_{j1}} v_{22}^{s_{j2}}) = (G_2^{s_{j1} + \beta_1 s_{j2}}, G_2^{\alpha_2 s_{j1} + \alpha_2\beta_2 s_{j2} - s_{j2}})$$

Solving for s_{j1} and s_{j2} we get $s_{j1} = k_j - \alpha_2\beta_2 k_j + \beta_2 l_j$ and $s_{j2} = \alpha_2 k_j - l_j$. The simulator can thus compute

$$G_2^{s_{j1}} = d_{j1}^{(1-\alpha_2\beta_2)} \cdot d_{j2}^{\beta_2} \qquad G_2^{s_{j2}} = d_{j1}^{\alpha_2} \cdot d_{j2}^{-1} \tag{7.7}$$

and use these values to compute ϕ, since it knows the logarithms of all v_{ij} as well as all r_{ij} and γ_{ij}, and θ, e.g.

$$\begin{aligned} v_{22}^{t_{22}} &= v_{22}^{\sum_{j=1}^n s_{j2} \sum_{i=1}^m r_{i2}\gamma_{ij}} = \big(G_2^{\sum_{j=1}^n s_{j2} \sum_{i=1}^m r_{i2}\gamma_{ij}}\big)^{\alpha_2\beta_2 - 1} = \textstyle\prod_{j=1}^n (G_2^{s_{j2}})^{(\alpha_2\beta_2 - 1)\sum_{i=1}^m r_{i2}\gamma_{ij}} \\ &= \big(\textstyle\prod_{j\in J} (d_{j1}^{\alpha_2} \cdot d_{j2}^{-1})^{(\alpha_2\beta_2 - 1)\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) \big(\textstyle\prod_{j\notin J} G_2^{s_{j2}(\alpha_2\beta_2 - 1)\sum_{i=1}^m r_{i2}\gamma_{ij}}\big) \ . \end{aligned}$$

Since $A_j = 1$ for $j \in J$, it is straightforward to compute θ. □

The above lemma lets us simulate a committed message, a committed signature and a proof of validity for a *given* committed public key (since in $\mathrm{E}_{\widehat{A}''_S}$ given in (7.8) below, the (implicit) constant that is paired with Y is 1, thus the premise of the lemma is satisfied). To prove anonymity of our construction of a delegatable-credential scheme in Sect. 7.2 we moreover need to simulate a verifiably encrypted signature on a given committed message; this requires simulation of a proof for equation $\mathrm{E}_{\widehat{A}''_S}$, where the constant (H^{-1}) that is paired with the value whose commitment is given (M) is not trivial; however, its logarithm -1 is known to the simulator.

We give a strengthening of Lemma 8, where the A_j are of the form $G_1^{a_j}$ with a_j known to the simulator.

Lemma 9. *Let* E *be as in* (2.2) *with* $\mathfrak{t}_T = 1_T$ *and* $A_j = G_1^{a_j}$ *(with* a_j *known) for indices* $j \in J$. *Given commitments* $\mathbf{d}_j$ *for* $j \in J$, *we can simulate* $\mathbf{c}_1, \ldots, \mathbf{c}_n$ *and* $\mathbf{d}_j$ *for* $j \notin J$ *and a proof* π *for* E *and* $(\mathbf{c}_1, \ldots, \mathbf{c}_m, \mathbf{d}_1, \ldots, \mathbf{d}_m)$ *if we are given the simulation trapdoor sim for* ck^*. *A symmetric result holds for* $\mathbf{c}_i$ *and* $\mathbf{d}_j$ *interchanged, and* $A_j = G_1^{a_j}$ *replaced with* $B_i = G_2^{b_i}$.

Proof. For simplicity, we give a proof in the additive notation of Remark 1 (p. 22). Since $\vec{X} = (0,\dots,0)^\top = \vec{Y}$, we have

$$\vec{\mathbf{c}} = R\vec{\mathbf{u}} \qquad \phi = R^\top \iota(\vec{B}) + (R^\top \Gamma S - Z^\top)\vec{\mathbf{v}}$$
$$\vec{\mathbf{d}} = S\vec{\mathbf{v}} \qquad \theta = S^\top \iota(\vec{A}) + Z\vec{\mathbf{u}}$$

Let us denote by $\vec{A}'$ the vector $\vec{A}$ where all A_j with $j \notin J$ are replaced by 0, and by $\vec{A}''$ the vector $\vec{A}$ where all A_j with $j \in J$ are replaced by 0, and let S' and S'' be defined analogously. We have $\vec{A} = \vec{A}' + \vec{A}''$ and $S = S' + S''$, and moreover $S^\top \iota(\vec{A}) = (S')^\top \iota(\vec{A}') + (S'')^\top \iota(\vec{A}'')$. Note that the simulator only knows the logarithms of $\vec{A}'$ (by the premise of the lemma) and the values in S'' (which it has chosen itself).

Since in the WI setting the matrix $\vec{\mathbf{u}}$ is invertible, there exists $\Omega \in \mathbb{Z}_p^{2\times 2}$ s.t. $\iota(\vec{A}') = \Omega\vec{\mathbf{u}}$. The logarithms of $\iota(\vec{A}')$ and $\vec{\mathbf{u}}$ being known to the simulator, it can efficiently compute Ω. We now show how the simulator computes the proof (ϕ, θ): it chooses $\widehat{Z} \leftarrow \mathbb{Z}_p^{2\times 2}$ and (knowing the values in S'') computes $\theta := (S'')^\top \iota(\vec{A}'') + \widehat{Z}\vec{\mathbf{u}} = S^\top \iota(\vec{A}) - (S')^\top \iota(\vec{A}') + \widehat{Z}\vec{\mathbf{u}}$, which is a proof $\theta = S^\top \iota(\vec{A}) + Z\vec{\mathbf{u}}$ with $Z := \widehat{Z} - (S')^\top \Omega$. With this randomness Z the first part of the proof is then

$$\phi = R^\top \iota(\vec{B}) + (R^\top \Gamma S - \widehat{Z}^\top + \Omega^\top S')\,\vec{\mathbf{v}} \ ,$$

which can be constructed using the techniques of the proof of Lemma 8: it suffices to construct the elements in (7.7) and use the known logarithms of $\vec{\mathbf{v}}$ as well as the known values R and Ω. □

7.4.2 Making the Equations for Ver'' Simulatable

In our application to delegatable credentials in Sect. 7.2 we have to simulate proofs for the equations of $\mathrm{E}_{\mathsf{Ver}''(\cdot,v,\cdot,\cdot)}(vk, (M,N), \Sigma)$, *when the commitments for* $vk = (X,Y)$ *or* (M,N) *(or both!) are given to the simulator.*

While E_B and E_R from (5.5) have a trivial right-hand side, we replace $\mathrm{E}_{A''}$ from (7.5) by the equations

$$\mathrm{E}_{A''_S}(A;\, W,S,N,D) \ : \ e(K\cdot L^v, \underline{W})\, e(T^{-1}, \underline{S})\, e(G^{-1}, \underline{N})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = 1$$
$$\mathrm{E}_{d_S}(d;\, W) \ : \ W^d \cdot H^{-d} = 1$$

First, we transformed $\mathrm{E}_{A''}$ into a homogeneous equation and a multi-scalar multiplication equation as described by [GS08]. We chose to turn H into a variable W, since proofs for equations in $\mathbb{G}_2$ are in $\mathbb{G}_1^4 \times \mathbb{G}_2^2$ and thus smaller than proofs for equations in $\mathbb{G}_1$; more importantly, this allows to use the variable W for several (formerly) inhomogeneous equations with a right-hand side $e(\cdot, H)$. For delegatable credentials this means that we can use the commitments to W and d, as well as the proof for E_{d_S} for *all* levels of the credential.

Moreover, in $\mathrm{E}_{A''_S}$ we replaced $e(M, H^{-1})$ by $e(G^{-1}, N)$ which by $\mathrm{E}_M(M;N)$ is equal. Accordingly, we replace E_U by

$$\mathrm{E}_{U_S}(Q,N) \ : \ e(T^{-1}, \underline{Q})\, e(G^{-1}, \underline{N}) = e(U,H)^{-1}$$

which, together with E_M and E_P proving that $(M,N) \in \mathcal{DH}$ and $(P,Q) \in \mathcal{DH}$, still asserts that $U = T^t \cdot M$. As for $\mathrm{E}_{A''}$ and E_U we again have that the left-hand side of E_{U_S} is contained in that of $\mathrm{E}_{A''_S}$. Note that in addition, this equation is *linear* in the sense of Groth-Sahai and a proof π_{U_S} thus reduces to two elements from $\mathbb{G}_1$, whereas $\pi_U \in \mathbb{G}_1^4 \times \mathbb{G}_2^4$.

Next, we show how to adapt SigCom'' (which is the algorithm in Figure 5.1 in Sect. 5.4.2 with A defined as $(K \cdot L^v \cdot T^r \cdot U)^{\frac{1}{x+c}}$ at the beginning). All that needs to be done to define SigCom''_S is to replace $\mathrm{E}_{\mathcal{A}^\dagger}$ by

$$\mathrm{E}_{A^\dagger_S}(A;\, W, D) \;:\; e(K \cdot L^v, \underline{W})\, e(\underline{A}, Y)\, e(\underline{A}, \underline{D}) = 1_T \;\; .$$

Setting $\pi'_{A_S} \leftarrow \pi_{U_S} \circ \mathsf{Prove}(ck, \mathrm{E}_{A^\dagger_S}, (A, \alpha), (W, \omega), (H^c, \delta))$ in Figure 5.1 yields thus a proof for $\mathrm{E}_{A''_S}$ by Lemma 5, since the product of the left-hand sides of $\mathrm{E}_{A^\dagger_S}$ and E_{U_S} is the left-hand side of $\mathrm{E}_{A''_S}$. The proof for the additional (multi-scalar multiplication) equation E_{d_S} can be produced by the signer herself. Finally, adapting $\mathsf{AdPrC}_{\mathcal{K}}$ to the simulatable equations is straightforward: it now adapts a proof for $\mathrm{E}_{A''_S}$ to one for $\mathrm{E}_{\widehat{A}''_S}$ (given in (7.8) below), where Y is a variable.

We demonstrated how our instantiation of commuting signatures based on $\mathbf{Sig}''$ can be adapted to make the equations for Ver'' simulatable. Below, we show that they can even be simulated when commitments to the verification key and/or the message are given to the simulator.

Simulating $\mathsf{Com}_{\mathcal{M}}$. When the simulator needs to simulate a $\mathsf{Com}_{\mathcal{M}}$ commitment it does the following: set $\mathbf{c}_M$ and $\mathbf{c}_N$ to commitments to 1. This enables simulation of other proofs for equations about M (such as those in Ver''). Since $\mathsf{Com}_{\mathcal{M}}$ also contains the value $U = T^t \cdot M$, the simulator has to choose t randomly, which defines P and Q. Now the simulator can produce $\mathbf{c}_P, \mathbf{c}_Q, \pi_M, \pi_P, \pi_U$ honestly. Note that the fact that $\mathbf{c}_P$ and $\mathbf{c}_Q$ were not produced as commitments to 1 is not a problem, as they are never used outside of a $\mathsf{Com}_{\mathcal{M}}$ commitment.

Simulating $\mathsf{Ver}''(\cdot,\cdot,\cdot)$ for Fixed Commitments. In the proof of anonymity of our credential scheme, we have to construct algorithms $\mathsf{SimCredProve}$ and $\mathsf{SimIssue}$ that output Groth-Sahai proofs without being given witnesses. The proofs are for validity of certificates contained in credentials, thus about the simulatable equations in Ver'' from Scheme 4 (p. 82). The only equation that contains parts of a verification key or the message of is the following

$$\mathrm{E}_{\widehat{A}''_S}(A;\, W, S, N, Y, D) \;:\; e(K \cdot L^v, \underline{W})\, e(T^{-1}, \underline{S})\, e(G^{-1}, \underline{N})\, e(\underline{A}, \underline{Y})\, e(\underline{A}, \underline{D}) = 1_T \;\; . \tag{7.8}$$

Since it satisfies the requirements for Lemma 9, we get:

Corollary 1. *Given commitments $\mathbf{c}_{vk}$ and $\mathbf{C}$, the simulator can produce $(\mathbf{c}_\Sigma, \widehat{\pi})$ that is distributed as*

$$\Big[\Sigma \leftarrow \mathsf{Sign}''(sk, v, (M, N));\; \rho \leftarrow \mathcal{R}_\Sigma \;:\; \big(\mathsf{Com}(ck^*, \Sigma, \rho),\; \mathsf{Prove}\big(ck^*, \mathrm{E}_{\mathsf{Ver}''(\cdot,v,\cdot,\cdot)}, ((X,\xi),(Y,\psi)), ((M,\mu),(N,\nu)), (\Sigma,\rho)\big)\big)\Big] \;\; ,$$

where $vk = (X, Y) \in \mathcal{DH}$ and (ξ, ψ) are such that $\mathbf{c}_{vk} = \mathsf{Com}(ck^*, vk, (\xi, \psi))$, sk is such that $vk = \mathsf{VK}(sk)$, and M, N, μ and ν are such that $\mathbf{C} = (\mathsf{Com}(ck^*, M, \mu), \mathsf{Com}(ck^*, N, \nu), \ldots)$.

Proof. Simulating a proof for Ver'' means simulating proofs for equations $\mathrm{E}_{\widehat{A}''_S}, \mathrm{E}_{d_S}, \mathrm{E}_B$ and E_R. The simulator makes commitments $\mathbf{c}_\Sigma$ to $(1, \ldots, 1)$. Proofs π_B and π_R are computed honestly and the first equation satisfies the premises of Lemma 9: the constants (the "A_j" in (2.2)) that are paired with N and Y are G^{-1} and 1, respectively, and thus have known logarithms. $\pi_{\widehat{A}''_S}$ can thus be simulated. π_{d_S} is simulated by opening $\mathbf{c}_d := \mathsf{Com}(ck^*, 1, 0)$ to 0, as described in [GS08]. □

Concluding, we have shown that when we replace $\mathrm{E}_{A''}$ by $\mathrm{E}_{A''_S}$ and E_{d_S} (and $\mathrm{E}_{\widehat{A}''}$ by $\mathrm{E}_{\widehat{A}''_S}$ and E_{d_S}), all equations for validity of a $\mathbf{Sig}''$ signature can be simulated, even if the commitments to the verification key and the message are fixed in advance. This shows that our instantiation of delegatable credentials satisfies the strong simulation-based anonymity definition by [BCC+09].

Conclusion

We introduced the concept of automorphic signatures and gave two instantiations; the first is based on previously introduced assumptions while the second is more efficient and can be instantiated in asymmetric bilinear groups. It relies on a new assumption, which we prove to hold in the generic group model. We used our scheme to give the first efficient instantiation of Fischlin's round-optimal blind signatures. Furthermore, we illustrated the numerous benefits of automorphic signatures by constructing fully-secure group signatures and anonymous credentials, and by giving the first efficient instantiation of anonymous proxy signatures, providing additional anonymity guarantees that have not been considered so far.

We leave as an open problem the construction of a practical automorphic signature whose messages are single group elements. It would also be interesting to see if the techniques used in Definition 2 can be generalized to vectors of arbitrary (but fixed) length; that is, to define a direct transformation from a signature scheme whose message space is a group to one signing an arbitrarily fixed number of messages.

We moreover defined and instantiated a new primitive we call *commuting signatures*. They allow users to encrypt different components of a triple consisting of a verification key, a message and a signature, and prove validity of the encrypted values. Most importantly, they enable signers that are given an encrypted message to produce an encryption of a signature on it together with a proof of validity.

We showed that this primitive enables the first instantiation of delegatable anonymous credentials with non-interactive issuing and delegation. Moreover, using our instantiation, the efficiency of the credential scheme improves significantly compared to the (only) previous instantiation. We believe that commuting signatures are an important tool in the construction of privacy-preserving primitives and that they will find further applications. In particular, we are confident that the approach to transferable fair e-cash from [4] can be revised using commuting signatures to obtain stronger anonymity guarantees.

List of Figures

Abbreviations

ADH-SDH	Asymmetric Double Hidden Strong Diffie-Hellman
APS	Anonymous proxy signature
AWF-CDH	Asymmetric Weak Flexible Computational Diffie-Hellman
BB-HSDH	Boneh-Boyen Hidden Strong Diffie-Hellman
BCCKLS	Belenkiy Camenisch Chase Kohlweiss Lysyanskaya Shacham
BSZ	Bellare Shi Zhang
CCA	Chosen-ciphertext attack
CDH	Computational Diffie-Hellman
CL	Camenisch Lysyanskaya
CMA	Chosen-message attack
CPA	Chosen-plaintext attack
CRS	Common reference string
DDH	Decisional Diffie-Hellman
DH-SDH	Double Hidden Strong Diffie-Hellman
DLIN	Decision Linear
EUF	Existential unforgeability
GE	Group element
GS	Groth Sahai
HSDH	Hidden Strong Diffie-Hellman
IND	Indistinguishability
KEA	Knowledge-of-Exponent Assumption
LHS	Left-hand side
LRSW	Lysyanskaya Rivest Sahai Wolf
MOS	Multi-originator signature
NIZK	Non-interactive zero-knowledge
NP	Non-deterministic polynomial-time
PKI	Public-key infrastructure
PoK	Proof of knowledge
PPE	Pairing-product equation
RHS	Right-hand side
SDH	Strong Diffie-Hellman
SXDH	Symmetric eXternal Diffie-Hellman
VES	Verifiably encrypted signature
WF-CDH	Weak Flexible Computational Diffie-Hellman
WI	Witness-indistinguishable
ZK	Zero-knowledge

Publications

[1] Georg Fuchsbauer and David Pointcheval. Anonymous proxy signatures. In Rafail Ostrovsky, Roberto De Prisco, and Ivan Visconti, editors, *SCN 08*, volume 5229 of *LNCS*, pages 201–217. Springer, September 2008.

[2] Georg Fuchsbauer and David Pointcheval. Anonymous consecutive delegation of signing rights: Unifying group and proxy signatures. In Véronique Cortier, Claude Kirchner, Mitsuhiro Okada, and Hideki Sakurada, editors, *Formal to Practical Security*, volume 5458 of *LNCS*, pages 95–115. Springer, 2009.

[3] Georg Fuchsbauer and David Pointcheval. Proofs on encrypted values in bilinear groups and an application to anonymity of signatures. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 132–149. Springer, August 2009.

[4] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, December 2009. Full version available at `http://eprint.iacr.org/2009/146`.

[5] Georg Fuchsbauer, Jonathan Katz, and David Naccache. Efficient rational secret sharing in standard communication networks. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 419–436. Springer, 2010.

[6] Georg Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009. `http://eprint.iacr.org/`. To appear as part of M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo: Structure-Preserving Signatures and Commitments to Group Elements, in *CRYPTO 2010*, Springer 2010.

[7] Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS*, volume 6123 of *LNCS*, pages 218–235. Springer, 2010. Full version available at `http://eprint.iacr.org/2010/040`.

[8] Georg Fuchsbauer. Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials. Cryptology ePrint Archive, Report 2010/233, 2010. `http://eprint.iacr.org/`.

[9] Xavier Boyen, Céline Chevalier, Georg Fuchsbauer, and David Pointcheval. Strong cryptography from weak secrets. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 297–315. Springer, 2010.

[10] David Galindo, Benoît Libert, Marc Fischlin, Georg Fuchsbauer, Anja Lehmann, Mark Manulis, and Dominique Schröder. Public-key encryption with non-interactive opening: New constructions and stronger definitions. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 333–350. Springer, 2010.

[11] Georg Fuchsbauer and Damien Vergnaud. Fair blind signatures without random oracles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 2010*, volume 6055 of *LNCS*, pages 16–33. Springer, 2010.

Bibliography

[ACdM05] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID tags via insubvertible encryption. In Vijayalakshmi Atluri, Catherine Meadows, and Ari Juels, editors, *ACM CCS 05*, pages 92–101. ACM Press, November 2005.

[ACHM05] 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/`.

[AF96] Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 244–251. Springer, November 1996.

[AHO10] Masayuki Abe, Kristiyan Haralambiev, and Miyako Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive, Report 2010/133, 2010. `http://eprint.iacr.org/`.

[AO09] Masayuki Abe and Miyako Ohkubo. A framework for universally composable non-committing blind signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 435–450. Springer, December 2009.

[BB04] Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004.

[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.

[BCC+09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, August 2009. Full version available at `http://eprint.iacr.org/2008/428`.

[BCKL08] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 356–374. Springer, March 2008.

[BCKL09] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, August 2009.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *STOC*, pages 103–112. ACM Press, 1988.

[BFPW07] Alexandra Boldyreva, Marc Fischlin, Adriana Palacio, and Bogdan Warinschi. A closer look at PKI: Security and efficiency. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 458–475. Springer, April 2007.

[BGLS03] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, May 2003.

[BGR98] Mihir Bellare, Juan A. Garay, and Tal Rabin. Fast batch verification for modular exponentiation and digital signatures. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 236–250. Springer, May / June 1998.

[BHY09] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 1–35. Springer, April 2009.

[BN05] 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, August 2005.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, June 2003.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, volume 5209 of *LNCS*, pages 39–56. Springer, September 2008.

[BP04] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 273–289. Springer, August 2004.

[BPW03] Alexandra Boldyreva, Adriana Palacio, and Bogdan Warinschi. Secure proxy signature schemes for delegation of signing rights. Cryptology ePrint Archive, Report 2003/096, 2003. `http://eprint.iacr.org/`.

[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.

[Bra99] Stefan Brands. Rethinking public key infrastructure and digital certificates—building privacy. PhD thesis, Eindhoven Inst. of Tech., The Netherlands, 1999.

[BSZ05] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, February 2005.

[BW06] Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, May / June 2006.

[BW07] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 1–15. Springer, April 2007.

[CCS09] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Antoine Joux, editor, *EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 351–368. Springer, April 2009.

[CGH98] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited (preliminary version). In *30th ACM STOC*, pages 209–218. ACM Press, May 1998.

[CGS07] Nishanth Chandran, Jens Groth, and Amit Sahai. Ring signatures of sub-linear size without random oracles. In Lars Arge, Christian Cachin, Tomasz Jurdzinski, and Andrzej Tarlecki, editors, *ICALP 2007*, volume 4596 of *LNCS*, pages 423–434. Springer, July 2007.

[Cha82] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203, 1982.

[Cha85] David Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[CHK+06] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: Efficient periodic n-times anonymous authentication. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 06*, pages 201–210. ACM Press, October / November 2006.

[CKW04] Jan Camenisch, Maciej Koprowski, and Bogdan Warinschi. Efficient blind signatures without random oracles. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, pages 134–148. Springer, September 2004.

[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.

[CL02] Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, September 2002.

[CL04] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, August 2004.

[CL06] Melissa Chase and Anna Lysyanskaya. On signatures of knowledge. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 78–96. Springer, August 2006.

[CLY09] Julien Cathalo, Benoît Libert, and Moti Yung. Group encryption: Non-interactive realization in the standard model. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 179–196. Springer, December 2009.

[Cv91] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, April 1991.

[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.

[Dam92] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 445–456. Springer, August 1992.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lopez-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. Cryptology ePrint Archive, Report 2010/196, 2010. `http://eprint.iacr.org/`.

[DP92] Alfredo De Santis and Giuseppe Persiano. Zero-knowledge proofs of knowledge without interaction (extended abstract). In *33rd FOCS*, pages 427–436. IEEE Computer Society Press, October 1992.

[FGHP09] Anna Lisa Ferrara, Matthew Green, Susan Hohenberger, and Michael Østergaard Pedersen. Practical short signature batch verification. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 309–324. Springer, April 2009.

[Fia90] Amos Fiat. Batch RSA. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 175–185. Springer, August 1990.

[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.

[GL07] Jens Groth and Steve Lu. A non-interactive shuffle with pairing based verifiability. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 51–67. Springer, December 2007.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems. In *STOC*, pages 291–304, New York, NY, USA, 1985. ACM.

[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.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for NP. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 339–358. Springer, May / June 2006.

[GPS08] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, December 2006.

[Gro07] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, December 2007.

[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. Full version available at `http://eprint.iacr.org/2007/155`.

[GSW09a] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Groth–sahai proofs revisited. Cryptology ePrint Archive, Report 2009/599, 2009. `http://eprint.iacr.org/`.

[GSW09b] Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Practical zero-knowledge proofs for circuit evaluation. In Matthew G. Parker, editor, *IMA Int. Conf.*, volume 5921 of *Lecture Notes in Computer Science*, pages 469–494. Springer, 2009.

[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*, volume 4392 of *LNCS*, pages 323–341. Springer, February 2007.

[JLO97] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 150–164. Springer, August 1997.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 581–600. Springer, March 2006.

[KJP06] Sébastien Kunz-Jacques and David Pointcheval. About the security of MTI/C0 and MQV. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 156–172. Springer, September 2006.

[KTY07] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group encryption. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 181–199. Springer, December 2007.

[KZ06] Aggelos Kiayias and Hong-Sheng Zhou. Concurrent blind signatures without random oracles. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 49–62. Springer, September 2006.

[KZ08] Aggelos Kiayias and Hong-Sheng Zhou. Equivocal blind signatures and adaptive UC-security. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 340–355. Springer, March 2008.

[LRSW00] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, volume 1758 of *LNCS*, pages 184–199. Springer, August 2000.

[LV08] Benoît Libert and Damien Vergnaud. Multi-use unidirectional proxy re-signatures. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 08*, pages 511–520. ACM Press, October 2008.

[MRY04] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 171–190. Springer, February 2004.

[MUO96] Masahiro Mambo, Keisuke Usuda, and Eiji Okamoto. Proxy signatures for delegating signing operation. In *ACM CCS 96*, pages 48–57. ACM Press, March 1996.

[Nao03] Moni Naor. On cryptographic assumptions and challenges (invited talk). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 96–109. Springer, August 2003.

[NIS07] Recommendation for key management, special publication 800-57 part 1, NIST, 03/2007, 2007.

[Oka06] Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *LNCS*, pages 80–99. Springer, March 2006.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, May 1999.

[PS00] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[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.

[RS09] Markus Rückert and Dominique Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 17–34. Springer, August 2009.

[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.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 256–266. Springer, May 1997.

[SMP08] Jacob C. N. Schuldt, Kanta Matsuura, and Kenneth G. Paterson. Proxy signatures secure against proxy key exposure. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *LNCS*, pages 141–161. Springer, March 2008.

[TW05] Mårten Trolin and Douglas Wikström. Hierarchical group signatures. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP 2005*, volume 3580 of *LNCS*, pages 446–458. Springer, July 2005.

[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.