This Journal Version appears
in Journal of Computer Security, Volume 21, Number 5, pages 627–661. 2013.

Short Blind Signatures

Olivier Blazy[1], Georg Fuchsbauer[2], David Pointcheval[3], and Damien Vergnaud[3]

[1] Ruhr-Universität Bochum, Germany
[2] Institute of Science and Technology, Austria
[3] École normale supérieure, Paris, France †

Abstract Blind signatures allow users to obtain signatures on messages hidden from the signer; moreover, the signer cannot link the resulting message/signature pair to the signing session. This paper presents blind signature schemes, in which the number of interactions between the user and the signer is minimal and whose blind signatures are short. Our schemes are defined over bilinear groups and are proved secure in the common-reference-string model without random oracles and under standard assumptions: CDH and the decision-linear assumption. (We also give variants over asymmetric groups based on similar assumptions.) The blind signatures are Waters signatures, which consist of 2 group elements.
Moreover, we instantiate partially blind signatures, where the message consists of a part hidden from the signer and a commonly known public part, and schemes achieving perfect blindness. We propose new variants of blind signatures, such as *signer-friendly* partially blind signatures, where the public part can be chosen by the signer without prior agreement, 3-party blind signatures, as well as blind signatures on multiple aggregated messages provided by independent sources.
We also extend Waters signatures to non-binary alphabets by proving a new result on the underlying hash function.

Keywords. Blind Signatures, E-Cash, WSN, Mixed Commitments.

1 Introduction

Blind signature schemes were proposed by Chaum in 1982 [10]. They define an interactive signature protocol between a user and a signer, guaranteeing that the signed message, and even the resulting signature, are unknown to the signer; this property is called *blindness*. More precisely, if the signer runs several executions of the protocol leading to several message/signature pairs, he cannot link a pair to a specific execution: the view of the signer is unlinkable to the resulting message/signature pair. This unlinkability can be either computational, in which case we talk about *computational blindness*, or information-theoretic, we then talk about *perfect blindness*. The second security property for blind signatures is a notion of unforgeability, which has been formalized by Pointcheval and Stern [25] motivated by the use of blind signatures for e-cash: a user should not be able to produce more message/signature pairs (coins) than the number of signing executions with the bank (withdrawals). More recently, Schröder and Unruh [26] revisited the security model for other contexts.

There have been several constructions with highly interactive blind-signing protocols (like [24]), before Fischlin [15] gave a generic construction of *round-optimal* blind signature schemes, where there is only one round of communication between the user and the signer. Abe et al. [1] have efficiently instantiated his blueprint, in which the user obtains a signature on the blinded message from the signer, but in order to achieve unlinkability, a blind signature is defined as a non-interactive zero-knowledge proof of knowledge of this signature, leading to an increase of the signature size.

In [5, 6], we presented a new approach, where instead of proving knowledge of it, the user merely *randomizes* the signature, which suffices to achieve unlinkability. Blind signatures are thus signatures of the underlying scheme, which are much shorter than proofs of knowledge thereof. In our construction, the underlying (and thus the blind) signatures are Waters signatures [28], which consist of 2 elements from a bilinear group, while the message is a scalar. In comparison, the most efficient scheme by Abe et al. [1] has messages consisting of two group elements, while a signature consists of 18+16 (in $\mathbb{G}_1$ and $\mathbb{G}_2$) group elements. Furthermore, our schemes are secure under the, meanwhile standard, "decision linear" (DLin) [8] assumption, while the schemes in [1] are based on newly introduced "q-type" assumptions. The drawback of our scheme is that, while being round-optimal, the user must send much more information to the signer during the blind signature-issuing protocol. While all round-optimal schemes mentioned so far are proven secure in the Common Reference String (CRS) model,

† DI/ENS (CNRS, ENS, INRIA)

round-optimal blind signatures without CRS have been proposed by Garg et al. [17], who however only give impractical generic constructions.

A loophole in standard blind signatures was first identified by Abe and Okamoto [3]: the signer has no control at all over which messages are signed. In classical e-cash schemes, unforgeability, which restricts a user's number of coins to the number of withdrawals, was sufficient. For the case that the bank wants to include an expiration date in the message (in addition to the user-chosen serial number), Abe and Fujisaki [2] propose *partially blind signatures*, where the user and the signer agree on part of the message before running the blind signing protocol.

The above-mentioned scheme from [1] was extended to partially blind signature scheme in [16]. More recently, Seo and Cheon [27] presented a construction leading to (partially) blind-signature schemes in the CRS model. However, their construction relies on a trick consisting in starting from prime-order groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_3$ and considering group elements in $\mathcal{G} = \mathbb{G}_1 \oplus \mathbb{G}_2 \oplus \mathbb{G}_3$. While their approach provides nice theoretical tools, the resulting signatures lie in $\mathcal{G}^2$ and are therefore three times longer than our proposal.

Our contributions. In this paper, we extend our earlier results from [5] in several directions. Instead of using an encryption scheme to blind the message to be signed, we use a *mixed commitment scheme* [13] in which commitments can be set up to either be perfectly binding (like encryption) or perfectly hiding.

We first present a blind signature scheme with perfect blindness, using the perfectly hiding setup of Groth-Sahai commitments [20]. We then extend the model of partially blind signatures to avoid prior agreement on the public part of the message between signer and user: signers can decide on its content only before sending their message in the signature-issuing protocol. (If the user does not agree with the public part, they can simply discard the signature and start anew.) We call this new primitive *signer-friendly partially blind signatures*. While of course not forbidding any prior agreement on the public part, this primitive offers an effectively round-optimal protocol (as there need not be any communication between the user and signer before running the protocol itself).

Using perfectly hiding commitments, we present a round-optimal signer-friendly partially blind signature with perfect blindness. Again, instead of having a computational overhead for the agreement, both signer and user can simply choose their contribution on the fly. The signer can always refuse to sign when the user's public information does not suit him, and the user can always choose to discard irrelevant signatures.

Using the perfectly binding setting for the mixed commitment (and thus discarding perfect blindness) we take advantage of the fact that user and signer can independently choose their inputs and consider a new context: the message to be signed is an aggregation of inputs that come from several independent sources which cannot communicate with each other. We consider several aggregation procedures. First we present a way to obtain a signature on the concatenation of the inputs and then present a shorter instantiation yielding a signature on the sum of the inputs messages. Addition of messages in often used, *e.g.* when counting votes, or aggregating sensor information, *etc.*

All our constructions are based on the Waters signature [28], which signs messages that are binary strings. We reconsider the programmable hash function used for these signatures over a *non-binary* alphabet, in a similar way to what Hofheinz and Kiltz [21] did for the binary case. We prove a negative result on the $(2, 1)$-programmability, but a positive one on the $(1, poly)$-programmability of this hash function. The latter result immediately yields Waters signatures over a non-binary alphabet, which in turn leads to a reduction in the number of public-key elements.

Instantiations. We give several instantiations of our different blind-signature schemes, all of which are based on weak assumptions. Our constructions mainly use the following two building blocks, from which they inherit their security: Groth-Sahai proofs for languages over pairing-friendly groups [20] and Waters signatures derived from the scheme in [28] and first used in [9]. Since verification of the revisited Waters signatures from [5] is a statement of the language for Groth-Sahai proofs, these two building blocks combine smoothly. Moreover, Waters signatures (and its variant) are fully randomizable thanks to the homomorphic property of its random coins. The first instantiations are over pairing-friendly elliptic curves and use linear commitments. Both unforgeability and blindness of these constructions rely solely on the decision linear assumption. An instantiation with improved efficiency, in *asymmetric*

bilinear groups, using the SXDH variant of Groth-Sahai proofs and commitments, is drafted in the Appendix A. This setting requires Waters signatures over asymmetric groups, which in [5] we proved secure under a slightly stronger assumption, termed CDH^+, which in symmetric groups is identical to CDH.

Applications. The properties of our blind signature schemes find various kinds of applications for anonymity:

E-voting. The security of several e-voting protocols (see [4, 11, 12]) relies on the fact that each ballot is certified by an election authority. Since this authority should not learn the votes, a blind signature scheme (or a partially blind one, if for example the authority wants to specify the election in the ballot) is usually used to achieve this property. Using a perfectly blind scheme, privacy is even achieved in an information-theoretic sense. Our scheme is the first to achieve this property without random oracles and under standard complexity assumptions.

E-cash. As mentioned above, partially blind signatures play an important role in many electronic-commerce applications. In e-cash systems, for instance, the bank issuing coins must ensure that the message contains accurate information such as the face value of the e-coin without seeing it. Moreover, in order to prevent double-spending, the bank's database has to record all spent coins. Partially blind signatures can cope with these requirements, since the bank can explicitly include some information such as the face value or the expiration date in the coin. The latter, for example, can be included in the coin by the bank without prior agreement with the client.

Data aggregation in networks. A wireless (ad hoc) sensor network (WSN) consists of many sensor nodes that are deployed for sensing the environment and collecting data from it. Since transmitting and receiving data are the most energy-consuming operations, data aggregation has been put forward as an essential paradigm in these networks. The idea is to combine the data coming from different sources, minimizing thus the number of transmissions and saving energy. In this setting a WSN usually consists of three types of nodes:

- *sensor nodes*, which are small devices equipped with one or more sensors, a processor and a radio transceiver for wireless communication;
- *aggregation nodes* (or aggregators) performing the data aggregation (*e.g.* average, sum, minimum or maximum of data); and
- *base stations*, responsible for querying the nodes and gathering the data collected by them.

WSNs are at high security risk and two important security goals when doing in-network data aggregation are *data confidentiality* and *data integrity*. When homomorphic encryption is used for data aggregation, end-to-end encryption allows aggregation of the encrypted data and thus end-to-end data confidentiality, as aggregators never decrypt any data. Achieving data integrity is a harder problem and the attack where a sensor node reports a false reading value is typically not considered (the impact of such an attack being usually limited). The main security threat is a *data-pollution attack* in which an attacker tampers with an aggregation node to make the base station receive wrong aggregated results.

While in most conventional data-aggregation protocols, data integrity and privacy are not preserved at the same time, our multi-source blind signature primitive yields data confidentiality and prevents data-pollution attacks simultaneously by using the following simple protocol:

1. Data aggregation is initiated by a base station, which broadcasts a query to the whole network.
2. Sensor nodes then report values of their readings which are encrypted under base station's public key to their aggregators.
3. The aggregators perform data aggregation via the homomorphic properties of the encryption scheme, (blindly) sign the result and route the aggregated results back to the base station.
4. The base station decrypts the aggregated data and the signature, which proves the validity of the gathered information to the base station (but also to any other third party).

2 Definitions

This section presents successively the various tools we need to describe the global framework and the security model for partially blind signature schemes. There are two different layers for the construction, one relying on commitments and the other on signatures.

2.1 Commitments

The original construction from [5] uses an encryption scheme $\mathcal{E}$ described via four polynomial-time algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ outputs a pair of keys, a (public) encryption key ek and a (private) decryption key dk;
- $\mathsf{Encrypt}(\mathsf{ek}, m; \rho)$ outputs an encryption c of the message m under the encryption key ek with randomness $\rho \in \mathcal{R}_e$;
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ outputs the plaintext m, encrypted in the ciphertext c or $\perp$ in case of an invalid ciphertext.

Such an encryption scheme is required to have the following security properties:

- *Correctness*: For every key pair $(\mathsf{ek}, \mathsf{dk})$ generated by KeyGen, every message m, and every random ρ, we should have
$$\mathsf{Decrypt}(\mathsf{dk}, \mathsf{Encrypt}(\mathsf{ek}, m; \rho)) = m \ .$$
- *Indistinguishability under Chosen-Plaintext Attack [18]*: This notion, formalized by the adjacent game, states that an adversary should not be able to efficiently guess which message has been encrypted even if he chooses the two candidate messages.

$\mathsf{Exp}^{\mathsf{ind}-b}_{\mathcal{E},\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND}, \mathsf{ek})$
4. $c^* \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS}, c^*)$
6. $\texttt{RETURN}\ b'$

We define the adversary's advantage as:
$$\mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{ind}-1}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind}-0}_{\mathcal{E},\mathcal{A}}(\lambda) = 1] \ .$$

To generalize the original construction from [5], we will replace encryption by a mixed commitment scheme [13]. A commitment scheme allows anyone to *commit* to a message m by running Commit on a key ck, m and some randomness ρ. The resulting commitment c should not reveal anything about the committed message, *i.e.*, the commitment is *hiding*. By revealing the randomness ρ, the commitment can be *opened*, as anyone can run $c' \leftarrow \mathsf{Commit}(\mathsf{ck}, m; \rho)$ and check whether $c' = c$. However, the commitment should be *binding*, in that for every c there is only a single m to which c can be opened (in that for all $m \neq m', \rho, \rho'$: $\mathsf{Commit}(\mathsf{ck}, m; \rho) \neq \mathsf{Commit}(\mathsf{ck}, m'; \rho')$).

In a *mixed commitment* scheme, the *commitment keys* ck can be set up in two ways, which are computationally indistinguishable. One type leads to *perfectly hiding* commitments, that is, the commitment does not contain any information about the committed value; whereas the second type leads to *perfectly binding* commitments, that is, not even an unbounded adversary can find a commitment and two different openings for it. We moreover require that there be a trapdoor which enables efficient extraction of the committed value.

Of course, when applied, one will use only one kind of setup depending on the main security goal or the properties demanded of the final scheme.

Definition 1 (Mixed Commitment Scheme [13]). A *mixed commitment scheme* $\mathcal{C}$ is a 5-tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{WISetup}, \mathsf{ExSetup}, \mathsf{Commit}, \mathsf{Extract})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param of the scheme, and more specifically the commitment key ck by running one of the following algorithms:

  - $\mathsf{WISetup}(1^\lambda)$, outputs a perfectly hiding commitment key ck;
  - $\mathsf{ExSetup}(1^\lambda)$, outputs a perfectly binding commitment key ck and an *extraction key* xk (to be kept secret);
- $\mathsf{Commit}(\mathsf{ck}, m; \rho)$, outputs a commitment c of a message m under a commitment key ck and a random $\rho \xleftarrow{\$} \mathbb{Z}$;
- $\mathsf{Extract}(\mathsf{xk}, c)$, if c is a commitment under a binding key ck, created together with xk, it outputs m.

We sometimes write $\mathsf{Decommit}(\mathsf{ck}, c, m; \rho)$, which verifies correct opening of a commitment by checking whether $c = \mathsf{Commit}(\mathsf{ck}, m; \rho)$.

We require that the two setups be indistinguishable, that is, given a commitment key ck, it should be hard to decide whether the key has been generated by $\mathsf{WISetup}$ or $\mathsf{ExSetup}$: for any polynomial-time adversary $\mathcal{A}$ receiving a key ck from either distribution, its advantage in distinguishing which distribution it was should be negligible in λ.

Note that by indistinguishability of the setups, perfectly hiding commitments are automatically computationally binding and perfectly binding commitments are automatically computationally hiding.

2.2 Signatures

We now review several signature primitives, from classical to blind signatures.

Definition 2 (Signature Scheme). A *signature scheme* $\mathcal{S}ig$ is a 4-tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the public (verification) key vk and the private (signing) key sk;
- $\mathsf{Sign}(\mathsf{sk}, m; s)$ on input a signing key sk, a message m from the message space $\mathcal{M}$ and random coins $s \in \mathcal{R}_s$, produces a signature σ.
- $\mathsf{Verify}(\mathsf{vk}, m, \sigma)$ checks whether σ is a valid signature on m, w.r.t. the public key vk; it outputs 1 if the signature is valid, and 0 otherwise.

We expect signatures to be *existentially unforgeable under chosen-message attacks* [19]: even after querying n valid signatures on chosen messages $\{m_i\}_{i=1}^n$, an adversary should not be able to output a valid signature on a new message.

$\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}(\lambda)$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk} : \mathtt{OSign}(\mathsf{sk}, \cdot))$
4. $b \leftarrow \mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5. `IF` $m^* \in \mathcal{SM}$ `THEN RETURN` 0
6. `ELSE RETURN` b

In $\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}$, the adversary has access to the following oracle:

- $\mathtt{OSign}(\mathsf{sk}, m)$ chooses $s \xleftarrow{\$} \mathcal{R}_s$ and returns $\mathsf{Sign}(\mathsf{sk}, m; s)$, after adding m to the set of signed messages $\mathcal{SM}$.

The probability of success in this game is denoted by

$$\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{euf}}_{\mathcal{S}ig,\mathcal{A}}(\lambda) = 1] \ .$$

Definition 3 (Randomizable Signature Scheme). Let $\mathcal{S}ig = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme with the following additional algorithm:

- $\mathsf{Random}(\mathsf{vk}, m, \sigma; s')$, on input a valid signature σ on a message m under vk, produces a new signature σ', again valid under vk on m, using the additional random coins $s' \in \mathcal{R}_s$.

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda)$
$\quad \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda);$
$\quad (\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param});$
$\quad \big((m_1,\sigma_1),\ldots,(m_{q_s+1},\sigma_{q_s+1})\big) \leftarrow \mathcal{U}^*(\mathsf{vk} : \langle \mathcal{S}(\mathsf{sk},\cdot),\cdot\rangle^{\leq q_s});$
$\quad \texttt{IF } \exists i \neq j : m_i = m_j \texttt{ OR } \exists i : \mathsf{Verify}(\mathsf{vk},m_i,\sigma_i) = 0 \texttt{ THEN RETURN } 0$
$\quad \texttt{ELSE RETURN } 1$

$\mathsf{Exp}^{\mathsf{bl}\text{-}b}_{\mathcal{BS},\mathcal{S}^*}(\lambda)$
$\quad \mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda);$
$\quad (\mathsf{vk},m_0,m_1,st_{\texttt{FIND}}) \leftarrow \mathcal{S}^*(\texttt{FIND},1^\lambda);$
$\quad b \leftarrow \{0,1\};$
$\quad st_{\texttt{ISSUE}} \leftarrow \mathcal{S}^*(\texttt{ISSUE},st_{\texttt{FIND}} : \langle\cdot,\mathcal{U}(\mathsf{vk},m_b)\rangle^{\leq 1},\langle\cdot,\mathcal{U}(\mathsf{vk},m_{1-b})\rangle^{\leq 1});$
$\quad \texttt{IF } \sigma_0 = \bot \texttt{ OR } \sigma_1 = \bot \texttt{ THEN } (\sigma_0,\sigma_1) \leftarrow (\bot,\bot);$
$\quad b^* \leftarrow \mathcal{S}^*(\texttt{GUESS},\sigma_0,\sigma_1,st_{\texttt{ISSUE}});$
$\quad \texttt{IF } b = b^* \texttt{ THEN RETURN } 1 \texttt{ ELSE RETURN } 0.$

Figure 1. Unforgeability and blindness for blind signatures

A signature scheme is called *randomizable* if for any $\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$, any message $m \in \mathcal{M}$, any random coins $s \in \mathcal{R}_s$, and $\sigma = \mathsf{Sign}(\mathsf{sk},m;s)$, the following two distributions are statistically indistinguishable:

$$\mathcal{D}_0 = \{s' \xleftarrow{\$} \mathcal{R}_s : \mathsf{Sign}(\mathsf{sk},m;s')\} \qquad \mathcal{D}_1 = \{s' \xleftarrow{\$} \mathcal{R}_s : \mathsf{Random}(\mathsf{vk},m,\sigma;s')\}$$

The usual unforgeability notions apply (except strong unforgeability, since the signature is malleable, by definition).

Definition 4 (Blind Signature Scheme). A *blind signature scheme* $\mathcal{BS}$ is a 4-tuple of polynomial-time algorithms/protocols $(\mathsf{Setup},\mathsf{KeyGen},\langle\mathcal{S},\mathcal{U}\rangle,\mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param.
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk},\mathsf{sk})$;
- Signature Issuing is an interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk},m)$, for a message $m \in \mathcal{M}$. It generates an output σ for the user: $\sigma \leftarrow \langle\mathcal{S}(\mathsf{sk}),\mathcal{U}(\mathsf{vk},m)\rangle$.
- $\mathsf{Verify}(\mathsf{vk},m,\sigma)$ outputs 1 if the signature σ is valid w.r.t. m and vk, 0 otherwise.

The security of a blind signature scheme is defined through two different notions [25], unforgeability and blindness (see Figure 1). An adversary $\mathcal{U}^*$ against the unforgeability tries to generate $q_s + 1$ valid signatures after at most q_s complete interactions with the honest signer. The blindness condition protects, on the other hand, against malicious signers. It states that a malicious signer $\mathcal{S}^*$ should be unable to decide which of two messages m_0, m_1 has been signed first in two executions (one for each message, hence the superscript "≤ 1" in $\mathsf{Exp}^{\mathsf{bl}\text{-}b}_{\mathcal{BS},\mathcal{S}^*}$ in Figure 1) with an honest user $\mathcal{U}$. Let σ_b be the signature on m_b. Note that the malicious signer $\mathcal{S}^*$ can choose the keys and thus the verification key vk given to users. However, if $\mathcal{S}^*$ refuses to sign one of the inputs (*i.e.* $\sigma_i = \bot$) then the two resulting signatures are set to $\bot$; the adversary therefore does not gain any advantage if he decides to prevent the normal game execution.

Our unforgeability notion slightly differs from the original one [25], in that we do not exclude malleability, as this could not be satisfied by the randomizable signatures we use. We thus count the number of distinct signed messages, which should not be larger than the number of interactions with the signer, whereas the original definition counted the number of distinct message/signature pairs: $\mathcal{BS}$ is *unforgeable* if, for any polynomial adversary $\mathcal{U}^*$ (malicious user), the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda)$ is negligible, where $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{U}^*}(\lambda) = 1]$, in the security game presented in Figure 1. In this experiment, the adversary $\mathcal{U}^*$ can interact q_s times with the signing oracle $\mathcal{S}(\mathsf{sk},\cdot)$ (hence the notation $\mathcal{U}^*(\mathsf{vk} : \langle\mathcal{S}(\mathsf{sk},\cdot),\cdot\rangle^{\leq q_s})$) to execute the blind signature protocol: the adversary should not be able to produce more signatures on distinct messages than interactions with the signer. Our relaxation from the original *One-More Forgery* security is to accommodate randomizable signatures, for which from a message/ signature pair one can generate many signatures on the same message. This is the same difference as between classical and strong existential unforgeability for signatures.

Definition 5 (Partially Blind Signature Scheme). A *partially blind signature scheme* $\mathcal{PBS}$ is a 4-tuple of polynomial-time algorithms and protocols $(\mathsf{Setup}, \mathsf{KeyGen}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$ generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$;
- Signature Issuing: this is an interactive protocol between $\mathcal{S}(\mathsf{sk}, \mathsf{info})$ and $\mathcal{U}(\mathsf{vk}, m, \mathsf{info})$, for a message $m \in \mathcal{M}$ and shared information info. It generates an output σ for the user:

$$\sigma \leftarrow \langle \mathcal{S}(\mathsf{sk}, \mathsf{info}), \mathcal{U}(\mathsf{vk}, m, \mathsf{info}) \rangle \ .$$

- $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{info}, \sigma)$ outputs 1 if the signature σ is valid with respect to the message $m \| \mathsf{info}$ and vk, and 0 otherwise.

The security requirements are a direct extension of the classical ones: for unforgeability, we consider $m \| \mathsf{info}$ instead of m, and for blindness, we condition the unlinkability between signatures with the same public part info. Without the latter restriction, the signer could simply distinguish which message was signed by comparing the public information. The unforgeability is strengthened by considering also the public information so that the signer can be sure that the user will not be able to exploit his signature in another context.

Signer-Friendly Partially Blind Signatures. An agreement on info can be a long and tedious process allowing both participant to launch a denial-of-service attack. Instead of considering a global info, we will split it into two parts $\mathsf{info}_c, \mathsf{info}_s$, one chosen by the user and one by the signer. While we dismiss the agreement part, we stress that a signer can refuse to sign a public information he does not like, and a user can refuse to use a signature on a public information he did not agree with, so this division does not weaken the requirement on the scheme.

Definition 6 (Signer-Friendly Partially Blind Signature Scheme). A *signer-friendly partially blind signature scheme* $\mathcal{PBS}$ is a 4-tuple of polynomial-time algorithms and protocols $(\mathsf{Setup}, \mathsf{KeyGen}, \langle \mathcal{S}, \mathcal{U} \rangle, \mathsf{Verify})$:

- $\mathsf{Setup}(1^\lambda)$ generates the global parameters param of the system;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$;
- Signature Issuing is an interactive protocol between $\mathcal{S}(\mathsf{sk}, \mathsf{info}_c, \mathsf{info}_s)$ and $\mathcal{U}(\mathsf{vk}, m, \mathsf{info}_c)$, for a message $m \in \mathcal{M}$, signer information info_s and common information info_c. It generates an output σ for the user: $\sigma \leftarrow \langle \mathcal{S}(\mathsf{sk}, \mathsf{info}_c, \mathsf{info}_s), \mathcal{U}(\mathsf{vk}, m, \mathsf{info}_c) \rangle$.
- $\mathsf{Verify}(\mathsf{vk}, m, \mathsf{info}_c, \mathsf{info}_s, \sigma)$ outputs 1 if the signature σ is valid with respect to the message $m \| \mathsf{info}_c \| \mathsf{info}_s$ and vk, and 0 otherwise.

We note that setting $\mathsf{info}_c := \mathsf{info}$ and $\mathsf{info}_s := \perp$ leads to a standard partially blind signature; whereas setting $\mathsf{info}_c = \mathsf{info}_s = \perp$ is the case of a standard blind signature. The signer always performs the last action in the signing protocol, and so if he does not want to sign a specific info, he can simply abort the protocol several times until the shared part suits his will, hence the name *signer-friendly*. This is why in the following we simply let him choose this input. If the user wants a specific word in the final message he can always add it to the blinded message. Intuitively this strengthens the unforgeability notion as the adversary (the user in this case) will not be able to choose the whole messages to be signed because of info_s. This is ensured in the security game, because the adversary must output valid signatures, therefore they should be done with the chosen info_s. For the blindness property, the adversary must choose two messages with the same public $\mathsf{info}_c \| \mathsf{info}_s$ component.

Security Games for Signer-Friendly Partially Blind Signatures. $\mathcal{PBS}$ satisfies *blindness* if, for any polynomial adversary $\mathcal{S}^*$ (malicious signer), the advantage $\mathsf{Succ}^{\mathsf{bl}\text{-}b}_{\mathcal{PBS},\mathcal{S}^*}(\lambda)$ is negligible, where

$$\mathsf{Succ}^{\mathsf{bl}}_{\mathcal{PBS},\mathcal{S}^*}(\lambda) = |\Pr[\mathsf{Exp}^{\mathsf{bl}}_{\mathcal{PBS},\mathcal{S}^*}(\lambda) = 1] - 1/2| \ ,$$

in the security game presented in Figure 2. If $\mathcal{S}^*$ refuses to sign one of the inputs (*i.e.* $\sigma_i = \perp$) then the two resulting signatures are set to $\perp$, therefore $\mathcal{S}^*$ does not gain any advantage if he decides to prevent the game execution. We let $\mathcal{S}^*$ choose both pieces of the public information, which corresponds to

$\mathsf{Exp}^{\mathsf{bl}\text{-}b}_{\mathcal{PBS},\mathcal{S}^*}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{vk}, m_0, m_1, st_{\texttt{FIND}}, \mathsf{info}_c, \mathsf{info}_s) \leftarrow \mathcal{S}^*(\texttt{FIND}, 1^\lambda)$;
$b \leftarrow \{0,1\}$;
$st_{\texttt{ISSUE}} \leftarrow \mathcal{S}^*(\texttt{ISSUE}, st_{\texttt{FIND}} : \langle\cdot, \mathcal{U}(\mathsf{vk}, m_b, \mathsf{info}_c)\rangle^{\leq 1}, \langle\cdot, \mathcal{U}(\mathsf{vk}, m_{1-b}, \mathsf{info}_c)\rangle^{\leq 1})$;
$\texttt{IF } \sigma_0 = \bot \texttt{ OR } \sigma_1 = \bot, (\sigma_0, \sigma_1) \leftarrow (\bot, \bot)$;
$b^* \leftarrow \mathcal{S}^*(\texttt{GUESS}, \sigma_0, \sigma_1, st_{\texttt{ISSUE}})$;
$\texttt{IF } b = b^* \texttt{ THEN RETURN } 1 \texttt{ ELSE RETURN } 0$.

Figure 2. Blindness for Signer-Friendly Partially Blind Signatures

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda)$
$(\mathsf{param}) \leftarrow \mathsf{Setup}(1^\lambda)$;
$(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$;
$\big((m_i, \mathsf{info}_{c,i}, \mathsf{info}_{s,i}, \sigma_i)\big)_{i\in\{1,\ldots,q_s+1\}} \leftarrow \mathcal{U}^*(\mathsf{vk} : \langle\mathcal{S}(\mathsf{sk},\cdot),\cdot\rangle^{\leq q_s})$;
$\texttt{IF } \exists i \neq j : (m_i, \mathsf{info}_{c,i}, \mathsf{info}_{s,i}) = (m_j, \mathsf{info}_{c,j}, \mathsf{info}_{s,j})$
$\quad \texttt{OR } \exists i : \mathsf{Verify}(\mathsf{vk}, m_i, \mathsf{info}_{c,i}, \mathsf{info}_{s,i}, \sigma_i) = 0 \texttt{ THEN RETURN } 0$
$\texttt{ELSE RETURN } 1$

Figure 3. Unforgeability for Signer-Friendly Partially Blind Signatures

the case where, in the real world, the signer aborts as long as the user's public information does not suit him. As with regular partially blind signatures, the public information must be the same in both challenge messages to avoid a trivial attack. $\mathcal{PBS}$ is *unforgeable* if, for any polynomial adversary $\mathcal{U}^*$ (malicious user), the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda)$ is negligible, where

$$\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda) = \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{PBS},\mathcal{U}^*}(\lambda) = 1] \ ,$$

in the security game presented in Figure 3.

2.3 Efficient Instantiations of the Building Blocks

First, let us briefly sketch the basic building blocks: Groth-Sahai commitments, and a variation of the Waters signature. They both need a pairing-friendly environment $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ is an admissible, non-degenerated, bilinear map, for two groups $\mathbb{G}$ and $\mathbb{G}_T$, of prime order p, generated by g and $g_t = e(g,g)$ respectively. From the following descriptions, it is easily seen that both schemes are randomizable.

Groth-Sahai Commitments. Groth and Sahai [20] proposed non-interactive zero-knowledge proofs of satisfiability of certain equations over bilinear groups. Using as witness group elements (and scalars) which satisfy the equation, the prover starts with making commitments on them. The commitment key is of the form $(\mathbf{u}_1 = (u_{1,1} = g^{x_1}, 1, g), \mathbf{u}_2 = (1, u_{2,2} = g^{x_2}, g), \mathbf{u}_3 = (u_{3,1}, u_{3,2}, u_{3,3})) \in (\mathbb{G}^3)^3$. Depending on the definition of $\mathbf{u}_3$, this commitment can be either perfectly hiding or perfectly binding.

– To commit a group element $X \in \mathbb{G}$, one chooses three random scalars $r_1, r_2, r_3 \in \mathbb{Z}_p$ and sets (where $\odot$ denotes component-wise multiplication)

$$\begin{aligned}\mathcal{C}(X) &:= (1,1,X) \odot \mathbf{u}_1^{r_1} \odot \mathbf{u}_1^{r_2} \odot \mathbf{u}_3^{r_3} \\ &= (c_1 = u_{1,1}^{r_1} \cdot u_{3,1}^{r_3}, c_2 = u_{2,2}^{r_2} \cdot u_{3,2}^{r_3}, c_3 = X \cdot g^{r_1+r_2} \cdot u_{3,3}^{r_3}) \\ &= (c_1 = g^{x_1 r_1} \cdot u_{3,1}^{r_3}, c_2 = g^{x_2 r_2} \cdot u_{3,2}^{r_3}, c_3 = X \cdot g^{r_1+r_2} \cdot u_{3,3}^{r_3})\end{aligned}$$

– To commit a scalar $x \in \mathbb{Z}_p$, one chooses two random scalars $\gamma_1, \gamma_2 \in \mathbb{Z}_p$ and sets

$$\begin{aligned}\mathcal{C}'(x) &:= (1,1,g)^x \odot \mathbf{u}_1^{\gamma_1} \odot \mathbf{u}_3^{x+\gamma_2} \\ &= (c'_1 = u_{3,1}^{x+\gamma_2} \cdot u_{1,1}^{\gamma_1}, c'_2 = u_{3,2}^{x+\gamma_2}, c'_3 = u_{3,3}^{x+\gamma_2} \cdot g^{x+\gamma_1}) \\ &= (c'_1 = u_{3,1}^{x+\gamma_2} \cdot g^{x_1\gamma_1}, c'_2 = u_{3,2}^{x+\gamma_2}, c'_3 = u_{3,3}^{x+\gamma_2} \cdot g^{x+\gamma_1}).\end{aligned}$$

The idea is that with a regular initialization of the commitment parameters ($\mathbf{u}_3 = \mathbf{u}_1^{\nu} \odot \mathbf{u}_2^{\mu}$, for two random scalars $\nu, \mu \in \mathbb{Z}_p$), these commitments are perfectly binding. The committed group elements can even be extracted if one knows x_1, x_2: $c_3/(c_1^{1/x_1} c_2^{1/x_2}) = X$, and $c'_3/({c'_1}^{1/x_1} {c'_2}^{1/x_2}) = g^x$.

However, if $\mathbf{u}_3$ is defined as $\mathbf{u}_3 = \mathbf{u}_1^{\nu} \odot \mathbf{u}_2^{\mu} \odot (1, 1, g^{-1}) = (u_{3,1} = u_{1,1}^{\nu}, u_{3,2} = u_{2,2}^{\mu}, u_{3,3} = g^{\nu+\mu-1})$, for two random scalars $\nu, \mu \in \mathbb{Z}_p$, the commitments are perfectly hiding. In addition, the two parameter initializations are indistinguishable under the DLin assumption. This is thus a mixed commitment.

To prove satisfiability of an equation (which is the statement of the proof), a Groth-Sahai proof uses these commitments and shows that the committed values satisfy the equation. The proof consists again of group elements and is verified by a pairing equation derived from the statement. In the perfectly binding setting the proof is perfectly sound, whereas in the perfectly hiding setting the proof perfectly hides the used witness.

Waters Signature. The *Waters signature scheme* was formally described in [28]. It was proved existentially unforgeable against chosen-message attacks under the CDH assumption.

- $\mathsf{Setup}(1^\lambda)$: The scheme is defined over a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. The parameters are a randomly chosen generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $(u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$ defining the function $\mathcal{F}\colon \{0,1\}^k \to \mathbb{G}$, $\mathcal{F}(M) = u_0 \prod u_i^{M_i}$. We set $\mathsf{param} := (p, \mathbb{G}, \mathbb{G}_T, e, g, h, (u_0, \ldots, u_k))$;
- $\mathsf{KeyGen}(\mathsf{param})$: Choose a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines $\mathsf{vk} = Y = g^y$, and $\mathsf{sk} = Z = h^y$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$: To sign a message $M \in \{0,1\}^k$, choose $s \xleftarrow{\$} \mathbb{Z}_p$ and define $\sigma = \big(\sigma_1 = Z \cdot \mathcal{F}(M)^s, \sigma_2 = g^s\big)$;
- $\mathsf{Verify}(\mathsf{vk} = Y, M, \sigma)$: Check whether $e(g, \sigma_1) \stackrel{?}{=} e(Y, h) \cdot e(\mathcal{F}(M), \sigma_2)$.

We also use another useful result on the Waters signature (as used in [22]):

Property 7 (Randomizability). The Waters signature scheme is randomizable: for a valid pair (M, σ), if we define $\sigma' = (\sigma_1 \cdot \mathcal{F}(M)^{s'}, \sigma_2 \cdot g^{s'})$ for a random scalar s' then σ' is a random signature on M.

Proof. If the initial signature was generated with randomness s, the modified signature corresponds to the signature on M with random coins $s + s'$. Since this is perfectly random in group $\mathbb{Z}_p$, it leads to a random signature on M.

Suffixed Waters Signatures. Instead of signing one message, we will sign, with some additional parameters, a concatenation of 3 messages using Waters signatures:

$$m = M || \mathsf{info}_c || \mathsf{info}_s = (M_1, \ldots, M_\ell, \mathsf{info}_1, \ldots, \mathsf{info}_{k-\ell}) \in \{0,1\}^k$$

3 Signatures and Mixed Commitments

In [5] we presented a general framework for building extractable signatures on randomizable ciphertexts. Once a user has sent a ciphertext of a message m, and received a signature on this ciphertext, the framework proposes an algorithm SigExt which allows to recover the signature on the plaintext m, when one knows the decryption key. This property has also been strengthened to *strong extractability* where the initial user can recover the signature without possessing the decryption key if he has kept the randomness used in the initial ciphertext.

To achieve perfect blindness, we need to discard encryption and use a perfectly hiding commitment instead. This is where strong extractability becomes interesting, as we want the user to be able to recover the signature on the plaintext even when no decryption key exists.

3.1 Signatures on Mixed Commitments

We now define a scheme of signatures on mixed commitments. Note that this generalizes the existing definition of signatures on ciphertexts from [5].

Definition 8 (Signatures on Mixed Commitments). A *signature scheme on a mixed commitment* $\mathcal{SC}$ is a 7-tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{CKeyGen}, \mathsf{Commit}, \mathsf{Sign}, \mathsf{Decommit}, \mathsf{Verify})$:

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(\lambda)$
$(\mathsf{param}) \leftarrow \mathsf{Setup}(1^\lambda)$; $\mathcal{SM} := \emptyset$
$\{(\mathsf{ck}_i, \mathsf{xk}_i)\}_{i=1}^n \leftarrow \mathsf{CKeyGen}^n(\mathsf{param})$; $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$
$(c, \sigma) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{vk}, \mathsf{ck} : \mathsf{sign}(\mathsf{sk}, \cdot, \cdot))$;
$m \leftarrow \mathsf{Decommit}(\mathsf{xk}, \mathsf{vk}, c)$
IF $m = \perp$ OR $m \in \mathcal{SM}$ OR $\mathsf{Verify}(\mathsf{vk}, \mathsf{ck}, c, \sigma) = 0$ THEN RETURN 0
RETURN 1

Figure 4. Unforgeability of Signatures on Mixed Commitments

- $\mathsf{Setup}(1^\lambda)$, where λ is the security parameter, generates the global parameters param for the associated encryption and signature schemes;
- $\mathsf{CKeyGen}(\mathsf{param})$ generates a commitment key ck and possibly the associated extraction key xk (according to the setting, *i.e.* whether $\mathsf{WISetup}$ or $\mathsf{ExSetup}$ was used);
- $\mathsf{SKeyGen}(\mathsf{param})$ generates a pair of keys, the verification key vk and the signing key sk;
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{vk}, m; r)$ produces a commitment c of $m \in \mathcal{M}$ under ck, using the random coins $r \in \mathcal{R}_c$. This commitment is intended to be later signed under the signing key associated to the verification key vk (the argument vk can be empty if the signing algorithm is universal and does not require a ciphertext specific to the signer);
- $\mathsf{CSign}(\mathsf{sk}, \mathsf{ck}, c; s)$, produces a signature σ on a commitment c and a signing key sk, using the random coins $s \in \mathcal{R}_s$, or $\perp$ if c is not valid (w.r.t. ck, and possibly vk associated to sk);
- $\mathsf{Decommit}(\mathsf{ck}, c, r, m)$ decommits c into a plaintext m, by showing that c is a valid commitment to m under ck with randomness r.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{ck}, c, \sigma)$ checks whether σ is a valid signature on c, w.r.t. the public key vk. It outputs 1 if the signature is valid, and 0 otherwise (possibly because of an invalid commitment c, with respect to ck, and possibly vk);
- $\mathsf{Recover}(\mathsf{vk}, \mathsf{ck}, c, \sigma, r)$ recovers a signature on the initial plaintext m committed in c, valid under vk using randomness r used in the commitment c.

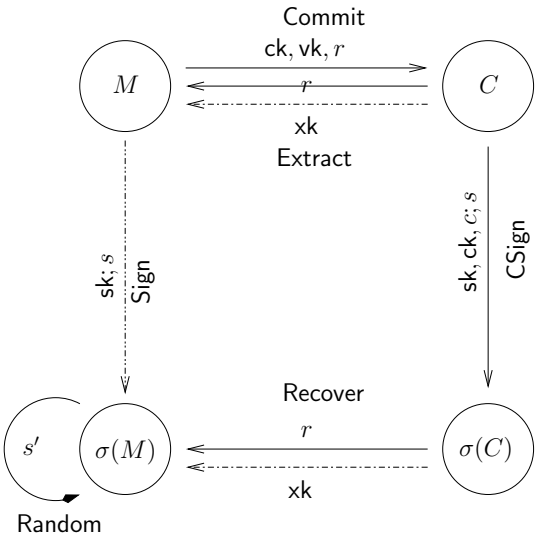
One could have additionally defined an $\mathsf{Extract}$ algorithm to recover the signature on the initial plaintext, or even the initial plaintext itself from the commitment, using the extraction key. But the latter will not always exist, whereas the random coins used for the commitment always exist.

Strong security notions for signatures on ciphertexts are not meaningful in our context, as we want signatures to be efficiently malleable, as long as the plaintext is not affected. Hence we further weaken the original definition of *existential unforgeability* (EUF) [19], where a new signature on an already signed message is not considered a forgery: In our definition a signature on a new ciphertext is not a forgery, if another ciphertext of the same plaintext has already been signed. Moreover, we require that the two setups for the mixed commitment be computationally indistinguishable. Under this assumption unforgeability in the perfectly binding setup also implies unforgeability in the perfectly hiding setting (in which we could not define the game).

We now define the unforgeability (UF) notion for signatures on mixed commitments, which makes sense in the perfectly binding setting only, but will help to prove some other security notions in any setting: $\mathcal{SC}$ is *unforgeable* if, for any polynomial-time adversary $\mathcal{A}$, the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(\lambda) = 1]$ is negligible, with $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}$ defined in Figure 4. In this experiment $\mathsf{sign}(\mathsf{sk}, \cdot, \cdot)$ is an oracle that takes as input a previously generated commitment key ck_i and a commitment c, runs CSign on it, and returns its output σ. It also adds to the set $\mathcal{SM}$ of signed plaintexts the plaintext $m = \mathsf{Extract}(\mathsf{xk}_i, \mathsf{vk}, c)$.

Unforgeability in the above sense thus states that no adversary is able to generate a new valid commitment-signature pair for a commitment to new message, *i.e.* different to those contained in commitments that were queried to the signing oracle.

A signature on mixed commitments with recovery provides the following: a user can commit to a message m and obtain a signature σ on the commitment c. Knowing the randomness used to compute c, from (c, σ) one can not only recover the committed message m, but also a signature σ' on the *message* m, using the functionality $\mathsf{Recover}$. The signature σ on the commitment c could thus be interpreted as a *commitment to a signature* on the message m. Commitment and signing can thus



A message M can be committed using random coins r.
The signer can sign this ciphertext.
A signature on the plaintext can be obtained using either the coins r (Recover), or with the extraction key xk if we are in the perfectly binding setup; the result is the same as a signature of M by the signer (Sign). This final signature can be randomized.

Figure 5. Recoverable Signatures on Mixed Commitments

be seen as commutative (see Figure 5). For completeness, we also define the Random algorithm for signatures, which we could also extend to signatures on committed values, but this is out of the scope of the applications this paper targets. Moreover, commitments must not be randomized if they are to be recovered later.

Figure 5 makes sense for any setup of the mixed commitment, while the unforgeability game in Figure 4 makes sense for the binding setup only.

Intuitively, the notion of blindness for our signatures comes from the hiding property of the commitment, because if the adversary can manage to find the order in which the message were signed then he has a bias in guessing which message is committed in each commitment (think of the final signature as a signature of a commitment). On the other hand, unforgeability of the blind signature is implied by unforgeability of the underlying signature, as we can recover the signature under the extraction property (now the final signature is a commitment to a signature), which relies on the binding property of the commitment.

3.2 Our Construction

Our approach combines Groth-Sahai commitments [20] and the Waters signature [28].

Assumptions. We rely on classical assumptions only: CDH for the unforgeability of signatures and DLin for the indistinguishability of the two commitment setups, which implies soundness of the proofs:

Definition 9 (The Computational Diffie-Hellman problem (CDH)). The CDH assumption, in a cyclic group $\mathbb{G}$ of prime order p, states that for a generator $g \in \mathbb{G}$ and random $a, b \in \mathbb{Z}_p$, given (g, g^a, g^b), it is hard to compute g^{ab}.

Definition 10 (Decision Linear Assumption (DLin)). The DLin assumption, in a cyclic group $\mathbb{G}$ of prime order p, states that given $(g, g^x, g^y, g^{xa}, g^{yb}, g^c)$ for random $a, b, x, y \in \mathbb{Z}_p$, it is hard to determine whether $c = a + b$ or a random value. When $(g, u = g^x, v = g^y)$ is fixed, a tuple (u^a, v^b, g^{a+b}) is called a linear tuple w.r.t. (u, v, g), whereas a tuple (u^a, v^b, g^c) for a random and independent c is called a random tuple.

One can easily see that if an adversary is able to solve a CDH challenge, then he can solve a DLin one. So the DLin assumption implies the CDH assumption.

Scheme. Let us now describe our signature on Groth-Sahai commitments:

– $\mathsf{Setup}(1^\lambda)$ chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. Moreover, it chooses an extra generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which defines the function $\mathcal{F}$. Setup returns $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \mathcal{F})$.

- $\mathsf{CKeyGen}(\mathsf{param})$ generates Groth-Sahai parameters $\mathsf{ck} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ in one of the two settings, and possibly the extraction key xk corresponding to the respective discrete logarithm in the binding setting.
- $\mathsf{SKeyGen}(\mathsf{param})$ chooses a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = Y = g^y$ and the secret key as $\mathsf{sk} = Z = h^y$.
- $\mathsf{Commit}(\mathsf{ck}, \mathsf{vk}, m; r)$: For a message $M \in \{0,1\}^k$ and random scalars $r = (r_1, r_2, r_3) \xleftarrow{\$} \mathbb{Z}_p^3$, defines the commitment as

$$c = \left(c_1 = u_{1,1}^{r_1} u_{3,1}^{r_3}, c_2 = u_{2,2}^{r_2} u_{3,2}^{r_3}, c_3 = g^{r_1+r_2} u_{3,3}^{r_3} \cdot \mathcal{F}(M)\right)$$

  and computes $Y_{1,2} = Y^{r_1+r_2}, Y_3 = Y^{r_3}$. It moreover generates Groth-Sahai proofs of consistency of the commitment:
  - A proof Π_M of knowledge of M in c, which consists of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k))$ and proofs that each committed value is a bit, as well as a proof that c is a commitment to $\mathcal{F}(M)$. Π_M is composed of $9k + 3$ group elements.
  - A proof Π_r containing the commitments $C_r = (\mathcal{C}(Y_{1,2}), \mathcal{C}(Y_3))$ asserting that they are correctly generated. This requires 9 group elements.

  Π thus consists of $9k + 12$ group elements.
- $\mathsf{CSign}(\mathsf{sk}, \mathsf{ck}, (c, \Pi); s)$: To sign the commitment c, this first checks if the proof Π is valid. If so, it outputs $\sigma = (Z \cdot {c_3}^s, u_{3,3}^s, g^s)$, for a random scalar $s \in \mathbb{Z}_p$.
- $\mathsf{Decommit}(\mathsf{ck}, c, r, m)$: Decommits c to m by simply showing that a commitment to m using ck and r is equal to c.
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{ck}, c, \sigma)$: Checks whether the following pairing equations are verified: $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(c_3, \sigma_3)$ and $e(\sigma_2, g) = e(u_{3,3}, \sigma_3)$.
- $\mathsf{Recover}(\mathsf{vk}, \mathsf{ck}, c, \sigma, r)$: If Verify is positive, one can use the randomness r to retrieve M and a valid signature on M:

$$\sigma' = (\sigma'_1 = \sigma_1 / (\sigma_3^{r_1+r_2} \sigma_2^{r_3}), \sigma'_2 = \sigma_3) \ ,$$

  which is a valid Waters signature.

Security Properties. In the next section, we generalize this construction to partially blind signatures and provide a security proof in any setting for the mixed commitment. Depending on the setting, we get either fair blind signatures (in the binding setting) or perfectly blind signatures when there is no shared public information.

Note that it suffices to do a security proof in one setting because of the indistinguishability of the two commitment setups. However, according to the setting, one will get perfect blindness or computational blindness.

4 Partially Blind Signatures

As in the previous section, our constructions will combine Groth-Sahai commitments [20] and Waters signatures [28] as follows: given a commitment on the "Waters hash" $\mathcal{F}(M)$ (and some additional values proving knowledge of the message M and the randomness used) and a public shared information info_c, the signer can make a partially blind signature on M, info_c and an extra piece of public information info_s.

4.1 Partially Blind Signatures with Perfect Blindness

With those building blocks, we design a partially blind signature scheme, where the user sends a commitment to the message and gets back a signature on it by the signer. Thanks to the random coins of the commitment, the user can "unblind" the received Waters signature. Finally, by randomizing it, the user breaks all links between the message/signature pair and the transaction.

Our protocol proceeds as follows, on a commitment of $F = \mathcal{F}(M)$, a public common message info_c, and a public message info_s chosen by the signer. It is split into five steps, that correspond to an optimal 2-flow protocol: Blind, which is first run by the user, CSign, which is then run by the signer, and Verify, $\mathsf{Unblind}$, Random, which are successively run by the user to generate the final signature. We thus have $\mathcal{U} = (\mathsf{Blind}; \mathsf{Verify}, \mathsf{Unblind}, \mathsf{Random})$ and $\mathcal{S} = \mathsf{CSign}$:

– $\mathsf{Setup}(1^\lambda)$ first chooses a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ and an additional generator $h \xleftarrow{\$} \mathbb{G}$. It generates a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which, for messages $M \in \{0,1\}^\ell$ and with k being the overall length of $M||\mathsf{info}_c||\mathsf{info}_s$, defines

$$\mathcal{F}_1(M) = u_0 \prod_{i=1}^{\ell} u_i^{M_i} \qquad \mathcal{F}_2(\mathsf{info}) = \prod_{i=\ell+1}^{k} u_i^{\mathsf{info}_i}$$

Moreover, it chooses Groth-Sahai parameters $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ in the perfectly hiding setting and outputs $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \mathcal{F}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$.

– $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = Y = g^y$ and the secret key as $\mathsf{sk} = Z = h^y$.

– Signature Issuing $\langle \mathcal{S}(\mathsf{sk}, \mathsf{info}_c, \mathsf{info}_s), \mathcal{U}(\mathsf{vk}, M, \mathsf{info}_c) \rangle$ is split into several steps:

- $\mathsf{Blind}(M, \mathsf{vk}; (r_1, r_2, r_3))$: For a message $M \in \{0,1\}^\ell$ and random scalars $(r_1, r_2, r_3) \xleftarrow{\$} \mathbb{Z}_p$, defines the commitment as

$$c = \left(c_1 = u_{1,1}^{r_1} u_{3,1}^{r_3}, c_2 = u_{2,2}^{r_2} u_{3,2}^{r_3}, c_3 = g^{r_1+r_2} u_{3,3}^{r_3} \cdot \mathcal{F}(M)\right)$$

  and computes $Y_{1,2} = Y^{r_1+r_2}, Y_3 = Y^{r_3}$. It also generates proofs of consistency:
  * A proof Π_M of knowledge of M in c, consisting of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_\ell))$, proofs that each committed value is a bit, and a proof that c commits to $\mathcal{F}(M)$. Π_M consists of $9\ell + 9$ group elements, as we have ℓ commitments, ℓ quadratic equations in $\mathbb{Z}_p$ of the type $M_i \cdot (1 - M_i) = 1$, and one quadratic multiscalar multiplication $u_0 \prod u_i^{M_i} = \mathcal{F}(M)$.
  * A proof Π_r containing the commitments $C_r = (\mathcal{C}(Y_{1,2}), \mathcal{C}(Y_3))$ asserting that they are consistent with c and C_M, *i.e.*, $e(c_3, Y) = e(g, Y_{1,2}) \cdot e(u_{3,3}, Y_3) \cdot e(\mathcal{F}(M), Y)$. This requires 9 additional group elements (6 for the commitments, and 3 for the Groth-Sahai proof).

  Π thus consists of $9\ell + 18$ group elements (when $\mathcal{M} = \{0,1\}^\ell$).
- $\mathsf{CSign}(\mathsf{sk}, (c, \Pi), \mathsf{info}_c, \mathsf{info}_s; s)$: To sign the commitment c, first check if the proof Π is valid. Then append the public message $\mathsf{info} = \mathsf{info}_c || \mathsf{info}_s$ to c_3 to create $c_3' = c_3 \cdot \mathcal{F}_2(\mathsf{info})$, which thus yields a commitment of the function evaluation on $M||\mathsf{info}_c||\mathsf{info}_s$ of global length k. Finally, output $\sigma = (Z \cdot (c_3')^s, u_{3,3}^s, g^s)$, for a random scalar $s \leftarrow \mathbb{Z}_p$, together with the additional public information info_s.
- $\mathsf{Verify}(\mathsf{vk}, (c, \mathsf{info}_c, \mathsf{info}_s), (\sigma_1, \sigma_2, \sigma_3))$: To check the validity of the signature, first compute $c_3' = c_3 \cdot \mathcal{F}_2(\mathsf{info})$ and then check whether the following pairing equations are satisfied:

$$e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(c_3', \sigma_3) \qquad e(\sigma_2, g) = e(u_{3,3}, \sigma_3)$$

  If not then σ is not a valid signature, and the user sets the blind signature as $\Sigma = \perp$.
- $\mathsf{Unblind}((r_1, r_2, r_3), \mathsf{vk}, (c, \mathsf{info}_c, \mathsf{info}_s), \sigma)$: If the previous verification is successful, use the random coins r_1, r_2, r_3 to recover a Waters signature on $M||\mathsf{info}_c||\mathsf{info}_s$: $\sigma' = (\sigma_1' = \sigma_1/(\sigma_3^{r_1+r_2}\sigma_2^{r_3}), \sigma_2' = \sigma_3)$.
- $\mathsf{Random}(\mathsf{vk}, (c, \mathsf{info}_c, \mathsf{info}_s), \sigma'; s')$: Randomize σ' to get the signature $\Sigma = (\sigma_1' \cdot \mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s)^{s'}, \sigma_2' \cdot g^{s'})$.

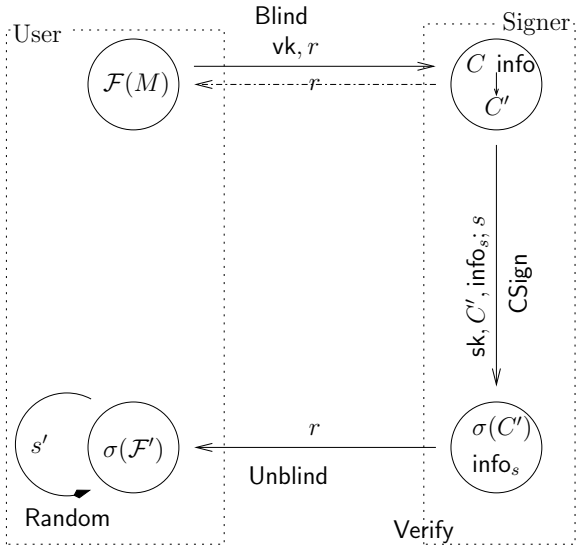
Note that Σ is a random Waters signature on $M||\mathsf{info}_c||\mathsf{info}_s$, where we denote $F = \mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s)$:

$$\begin{aligned}
\Sigma &= (\sigma_1' \cdot F^{s'}, \sigma_2' \cdot g^{s'}) = (F^{s'} \cdot \sigma_1/(\sigma_3^{r_1+r_2}\sigma_2^{r_3}), g^{s'} \cdot \sigma_3) \\
&= (F^{s'} \cdot Z \cdot {c_3'}^s/(g^{s(r_1+r_2)} u_{3,3}^{sr_3}), g^{s+s'}) \\
&= (F^{s'} \cdot Z \cdot g^{s(r_1+r_2)} u_{3,3}^{sr_3} \cdot F^s/(g^{s(r_1+r_2)} u_{3,3}^{sr_3}), g^{s+s'}) \\
&= (F^{s+s'} \cdot Z, g^{s+s'}) \;=\; \mathsf{Sign}(Z, M||\mathsf{info}_c||\mathsf{info}_s; s+s')
\end{aligned}$$

– $\mathsf{Verify}(\mathsf{vk}, (M, \mathsf{info}_c, \mathsf{info}_s), \Sigma = (\Sigma_1, \Sigma_2))$, is defined as Waters signature verification by checking that the following holds:

$$e(\Sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s), \Sigma_2) \ .$$

Theorem 11. *This signer-friendly partially blind signature scheme is unforgeable under the* DLin *assumption in* $\mathbb{G}$.



Blind hides a message M using random coins r. The signer can concatenate public messages info_c and info_s to the original commitment, which yields a commitment C' on $F = \mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s)$.
Using randomness r, Unblind recovers a signature on the plaintext, which is the same as a direct signature on $M||\mathsf{info}_c||\mathsf{info}_s$ by the signer.
Randomizing this signature prevents the signer to link the transaction to a signature.

Figure 6. Partially Blind Signatures with Perfect Blindness

Proof. Let us denote $\mathcal{PBS}$ our partially blind signature above. Assuming there is an adversary $\mathcal{A}$ against unforgeability that succeeds with probability ϵ, we build an adversary $\mathcal{B}$ against CDH, whose hardness is implied by that of DLin.

DLin *Assumption.* Breaking unforgeability means that after q_s interactions with the signer, the adversary manages to output $q_s + 1$ valid message/signature pairs on distinct messages. If the adversary $\mathcal{A}$ succeeds with probability ϵ when the commitment key is perfectly hiding, then $\mathcal{A}$ also succeeds with a probability negligibly close to ϵ when the commitment key is perfectly hiding. Otherwise $\mathcal{A}$ could be used to distinguish the two types of commitment keys, which are indistinguishable under DLin.

Signer simulation. Let us thus consider $\mathcal{PBS}$ but with a commitment scheme using the *binding* setting, say $\mathcal{PBS}'$. Our simulator $\mathcal{B}$ can thus extract values from the commitments since it will know the extraction key. We assume that $\mathcal{A}$ is able to break the unforgeability of $\mathcal{PBS}'$ with probability ϵ' after q_s interactions with the signer, based on which we build an adversary $\mathcal{B}$ against the CDH problem. (We basically follow Waters' original proof, but adapt it, since we need to simulate signatures $(\sigma_1, \sigma_2, \sigma_3)$ on commitments, rather than plain Waters signatures.)

Let $(A = g^a, B = g^b)$ be a challenge CDH-instance in a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$. We generate the global parameters using this instance: for simulating Setup/KeyGen, $\mathcal{B}$ picks a random position $j \xleftarrow{\$} \{0, \ldots, k\}$, chooses random indices $y_0, y_1, \ldots, y_k \xleftarrow{\$} \{0, \ldots, 2q_s - 1\}$, and random scalars $z_0, z_1, \ldots, z_k \xleftarrow{\$} \mathbb{Z}_p$. Define $Y = A = g^a$, $h = B = g^b$, $u_0 = h^{y_0 - 2jq_s} g^{z_0}$, and $u_i = h^{y_i} g^{z_i}$ for $i = 1, \ldots, k$. $\mathcal{B}$ also picks random scalars ν, μ, x_1, x_2, and generates binding Groth-Sahai parameters $(\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$ with $(\mathbf{u}_1 = (u_{1,1} = g^{x_1}, 1, g), \mathbf{u}_2 = (1, u_{2,2} = g^{x_2}, g), \mathbf{u}_3 = \mathbf{u}_1^{\nu} \odot \mathbf{u}_2^{\mu})$. Note that $u_{3,3} = g^{\nu+\mu}$. It outputs $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \mathcal{F}, \mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$. (The signing key is implicitly defined as $Z = h^a = B^a = g^{ab}$, which is the solution to our Diffie-Hellman instance.)

To answer a signing query for $(c = (c_1, c_2, c_3), \Pi)$, $\mathcal{B}$ first checks the proof $\Pi = (\Pi_M, \Pi_r)$. Using the commitment extraction key (x_1, x_2), it extracts M from the bit-by-bit commitments in Π_M and $Y_{1,2} = Y^{r_1+r_2}$, $Y_3 = Y^{r_3}$ from Π_r (where r_1, r_2, r_3 are the random coins of c). Furthermore, we can compute $c_3' = g^{r_1+r_2} u_{3,3}^{r_3} \cdot F$, where we denote $M' = M||\mathsf{info}_c||\mathsf{info}_s$ and $F = \mathcal{F}(M||\mathsf{info}_c||\mathsf{info}_s)$. $\mathcal{B}$ defines

$$H = -2jq_s + y_0 + \textstyle\sum_i y_i M_i' \qquad J = z_0 + \textstyle\sum_i z_i M_i' \qquad F = h^H g^J$$

If $H \equiv 0 \pmod p$ then $\mathcal{B}$ aborts, otherwise it sets

$$\sigma = (Y^{-J/H}(Y_{1,2} Y_3^{\nu+\mu})^{-1/H}(F(c_1^{1/x_1} c_2^{1/x_2}))^s, (Y^{-1/H} g^s)^{\nu+\mu}, Y^{-1/H} g^s).$$

Defining $\tilde{s} = s - a/H$, we have

$$\begin{aligned}\sigma_1 =& Y^{-J/H}(Y_{1,2}Y_3^{\nu+\mu})^{-1/H}(h^H g^J (c_1^{1/x_1} c_2^{1/x_2}))^s = Z \cdot (c_3')^{\tilde{s}}\\ \sigma_3 =& Y^{-1/H} g^s = Y^{-1/H} g^{\tilde{s}+a/H} = g^{\tilde{s}}\\ \sigma_2 =& (\sigma_3)^{\nu+\mu} = g^{(\nu+\mu)\tilde{s}} = u_{3,3}^{\tilde{s}}\end{aligned}$$

This thus exactly looks like a real signature sent by the signer.

Diffie-Hellman extraction. After at most q_s signing queries $\mathcal{A}$ outputs $q_s + 1$ valid Waters signatures. Since these are more than the number of signing queries, there is a least one message M^* that is different from all the messages $M||\mathsf{info}_c||\mathsf{info}_s$ involved in the signing queries. We define

$$H^* = -2jq_s + y_0 + \textstyle\sum_i y_i M_i^*, \qquad J^* = z_0 + \textstyle\sum_i z_i M_i^* \qquad \mathcal{F}(M^*) = h^{H^*} g^{J^*}$$

If $H^* \not\equiv 0 \pmod p$ then $\mathcal{B}$ aborts, otherwise, for some s^*, we have $\sigma^* = (h^a \mathcal{F}(M^*)^{s^*}, g^{s^*}) = (h^a g^{s^* J^*}, g^{s^*})$. Then, $\sigma_1^*/(\sigma_2^*)^{J^*} = h^a = g^{ab}$, which is a solution for the CDH problem.

Success probability. (this is based on [21]) The Waters hash function is $(1, q_s)$-programmable (*i.e.*, we can find with non negligible probability a case where q_s intermediate hashes H are non zero, and the last one is); therefore the previous simulation succeeds with non negligible probability $(\Theta(\epsilon/q_s\sqrt{k}))$, with which $\mathcal{B}$ then breaks CDH.

Theorem 12. *This signer-friendly partially blind signature scheme achieves perfect blindness.*

Proof. Since the commitment key is perfectly hiding, the transcript sent to the signer contains a commitment on the message to be signed which leaks no information about M. The additional proofs are perfectly witness-indistinguishable and thus do not provide any additional information about M. (For Groth-Sahai proofs in the perfectly hiding setting, for any M, committed with randomness r and any message M', there exists a random r' such that both commitment values collide.) Moreover, due to the perfect randomizability of Waters signatures, the output blind signature is uniformly random in the set of signatures on $M||\mathsf{info}_c||\mathsf{info}_s$, on which no information leaked. So the resulting signature is independent from the transcript seen by the signer.

5 Multi-Source Blind Signatures

5.1 Concatenation

The previous constructions enables a user to obtain a signature on a plaintext without revealing it to the signer. But what if the original message is coming from various users? We now present a new way to obtain a blind signature without requiring multiple users to combine their messages, providing once again a round-optimal way to achieve our goal.

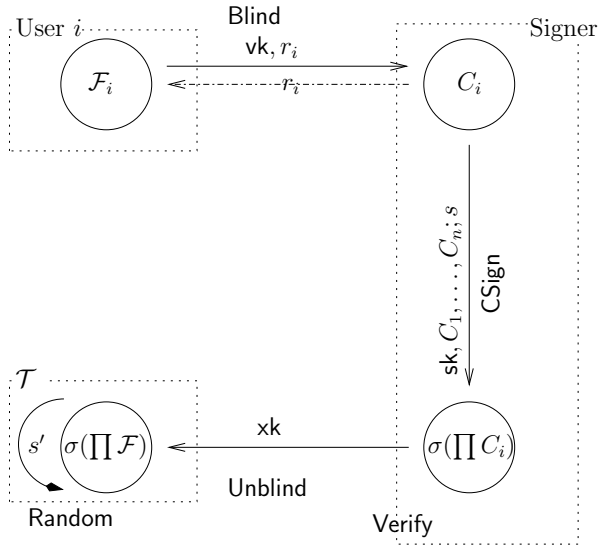
We thus consider another variant of our blind signature scheme. Setup no longer creates perfectly hiding parameters, but perfectly binding parameters. We therefore need not compute $u_{3,3}^s$ to run Unblind, since we can use the extraction key instead of the coins. In addition, in this scenario we do not consider a unique user providing a blinded message, but several users. The signer will produce a signature on a multi-source message, provided as different commitments. The signature and the messages will actually be committed under a key from a third party, which will be the only one able to extract the message and the signature.

Our instantiation is similar to the previous ones in the perfectly binding setting. For simplicity, we remove the partially blind part, but of course it could be adapted in the same way. With the previous building blocks, we will sign several commitments of $F_i = \mathcal{F}_i(M_i)$, and instead of the protocol $\langle \mathcal{U}, \mathcal{S} \rangle$, we now have one with three kind of participants: users $\mathcal{U}_i$ will blind a commitments on $\mathcal{F}_i(M_i)$, signer $\mathcal{S}$ signs the blinded message, and $\mathcal{T}$, the tallier, will verify, unblind and randomize this signature:

– $\mathsf{Setup}(1^\lambda)$: In a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$ the algorithm outputs a generator $h \xleftarrow{\$} \mathbb{G}$ and a vector

$$\boldsymbol{u} = \left(u_0, (u_{i,1}, \ldots, u_{i,\ell})_{i=1}^n\right) \xleftarrow{\$} \mathbb{G}^{n\ell+1} \ ,$$

where ℓ is a polynomial in λ. We define $\mathcal{F}_i(M_i) = \prod_\ell u_{i,\ell}^{m_{i,\ell}}$.



Different users can hide messages M_i using random coins r_i (Blind).
The signer can concatenate the messages inside the commitments, creating a commitment on $\mathcal{F} = \prod \mathcal{F}_i$.
Using the extraction key xk in Unblind, the tallier $\mathcal{T}$ can recover a signature on the plaintext, which is the same as a signature by the signer on the concatenation of all the messages M_i.
Randomizing this signature prevents the signer from linking the signature to a transaction.

Figure 7. Multi-Source Blind Signature on Concatenation

- $\mathsf{KeyGen}(\mathsf{param})$: Choose $y \xleftarrow{\$} \mathbb{Z}_p$, which defines $\mathsf{vk} = Y = g^y$ and $\mathsf{sk} = Z = h^y$, and generate a perfectly binding Groth-Sahai commitment key ck together with an extraction key $\mathsf{xk} = (x_1, x_2) \in \mathbb{Z}_p^2$.
- $(\mathcal{U}_i, \mathcal{S}, \mathcal{T})$:
  - $\mathsf{Blind}(M, \mathsf{vk}; (r_1, r_2, r_3))$ (we omit the subscripts i): For a message $M \in \{0,1\}^\ell$ and random scalars in $\mathbb{Z}_p$, define the commitment $c = \mathcal{C}(\mathcal{F}(M)) = (c_1, c_2, c_3)$. As before, we add proofs to this commitment:
    - A proof Π_M of knowledge of M, such that c commits to $\mathcal{F}(M)$, which consists of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k))$ and proofs that each committed value is a bit.
    - A proof Π_r containing the commitments $(\mathcal{C}(Y^{r_1+r_2}), \mathcal{C}(Y^{r_3}))$ together with proofs of consistency with c and C_M.
  - $\mathsf{CSign}(\mathsf{sk}, (c = (c_{1,i}, c_{2,i}, c_{3,i}), \Pi_i)_{i=1}^n; s)$: To sign n commitments, first check that all Π_i's are valid; after randomizing these commitments, compute the commitment $C = (\prod c_{1,i}, \prod c_{2,i}, u_0 \prod c_{3,i})$ and output $C = (C_1, C_2, C_3)$ and $\sigma = (C_1^s, C_2^s, Z \cdot C_3^s, g^s)$.
  - $\mathsf{Verify}(\mathsf{vk}, (C_1, C_2, C_3), (\sigma_1, \sigma_2, \sigma_3, \sigma_4))$: In order to check validity of the signature, verify the following equations: $e(\sigma_1, g) = e(C_1, \sigma_4)$, $e(\sigma_2, g) = e(C_2, \sigma_4)$, and $e(\sigma_3, g) = e(h, \mathsf{vk}) \cdot e(C_3, \sigma_4)$.
  - $\mathsf{Unblind}(\mathsf{xk}, \mathsf{vk}, c = (C_1, C_2, C_3), \Pi = (\Pi_i)_{i=1}^n, \sigma)$: From a valid signature σ on c, knowing the extraction key (x_1, x_2), one can extract the message M from the bit-by-bit commitments in Π. One can also extract the corresponding signature σ' as: $\sigma'_1 = \sigma_3 / (\sigma_1^{1/x_1} \sigma_2^{1/x_2})$, $\sigma'_2 = \sigma_4$, which is a Waters signature on M, the concatenation of all the messages.
  - $\mathsf{Random}(\mathsf{vk}, M, \sigma'; s')$: The signature σ' is randomized to get the blind signature $\Sigma = (\sigma'_1 \cdot \mathcal{F}(M)^{s'}, \sigma'_2 \cdot g^{s'})$.
- $\mathsf{Verify}(\mathsf{vk}, M, \sigma = (\sigma_1, \sigma_2))$: In order to check the validity of the signature, one checks whether: $e(\sigma_1, g) \stackrel{?}{=} e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

Theorem 13. *The above multi-source blind signature scheme for concatenation is blind and unforgeable under the* DLin *and* CDH *assumptions, respectively: no adversary can generate more message/signature pairs on distinct messages than the number of interactions with the signer.*

The theorem follows from the previous result, as the combination of the different partial Waters hashes can be seen as one global hash (note that we have independent generators for each index). The perfectly binding Groth-Sahai proofs guarantee that each user outputs a Waters hash of their message under their generators.

5.2 Linear Operations

The previous scheme presents a way to combine multiple blind messages into one by concatenation. One drawback is that every bit in every one of the concatenated messages requires a generator in the setup. Let us now assume all that is required is a signature on the sum of the messages (or the mean, or any other linear operation). Concatenation could still be used, as the verifier could perform the linear operation at the end, but all individual messages are signed, requiring a long public key. We improve the construction, reducing the public key size and the information leaked about the individual messages when one only requires a signature on the sum or the mean of the individual messages. Instead of signing the concatenation of the messages, we now consider the sum of messages, for which we can allow every user to use the same generators for the function $\mathcal{F}$, which reduces the public key by a factor of the number of aggregated messages.

The resulting scheme is the same as before except that for Setup we have $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$. We then proceed as before considering $\mathcal{F}(M_i) = \prod_\ell u_\ell^{m_{i,\ell}}$, which are aggregated to $\prod_i \mathcal{F}(M_i)$, which is $\mathcal{F}$ evaluated on the *sum* of all messages. However, the exponents in the Waters hash function are no longer bits but belong to a larger alphabet (*e.g.* $\{0, \ldots, t\}$ if t users participate and send bit strings). Following the work done in [21], we will show in the next section that over a non-binary alphabet the Waters function remains $(1, poly)$-programmable as long as the size of the alphabet is a polynomial in the security parameter. This result readily implies the security of the multi-source blind signature scheme for addition, but also any linear combination.

Theorem 14. *This multi-source blind signature scheme for addition is blind and unforgeable under the* DLin *assumption as long the alphabet size and the number of sources are polynomial in the security parameter.*

Note however that more than just the sum is leaked since carries are not propagated, but accumulated in each digit. Hence, the value of the sum in one digit leaks some information on the same digits on individual messages. Anyway, our goal is to authenticate the result with minimal communication.

6 Waters Function and Non-binary Alphabets

In this section we prove that for a polynomial-size alphabet, the Waters function remains programmable. We recall some notations introduced in [21] and show our result, which can be seen as an improvement over that presented by Naccache [23], who considered a variant of Waters identity-based encryption [28] with shorter public parameters.

6.1 Definitions

Let us recall some basic definitions. Considering a cyclic group $\mathbb{G}$, for some security parameter λ, we define a *group hash function* H for G, an alphabet $\Sigma = \Sigma(\lambda)$ and an input length $\ell = \ell(\lambda)$ as a pair of probabilistic polynomial-time algorithms $(\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ such that:

- $\mathsf{PHF.Gen}$ takes as input a security parameter λ and outputs a key κ
- $\mathsf{PHF.Eval}$ takes as input a key κ output by $\mathsf{PHF.Gen}$ and a string $X \in \Sigma^\ell$ and outputs an element of $\mathbb{G}_\lambda$.

Definition 15 ([21]). A group hash function $(\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ is (m, n, δ)-programmable, if there exist two PPT algorithms $(\mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval})$ such that:

- **Syntax:** For $g, h \in \mathbb{G}$, $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ generates a key κ' and a trapdoor t such that $\mathsf{PHF.TrapEval}(t, X)$ produces integers a_X, b_X for any $X \in \Sigma^\ell$.
- **Correctness:** For all $g, h \in \mathbb{G}$, all $(\kappa', t) \leftarrow \mathsf{PHF.TrapGen}(1^\lambda, g, h)$ and all $X \in \Sigma^\ell$, $H_{\kappa'}(X) := \mathsf{PHF.Eval}(\kappa', X)$ satisfies $H_{\kappa'}(X) = g^{a_X} h^{b_X}$ where $(a_X, b_X) := \mathsf{PHF.TrapEval}(t, X)$.
- **Statistically close trapdoor keys:** For all generators $g, h \in \mathbb{G}^2$, the functions $\mathsf{PHF.Gen}(1^\lambda)$ and $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ output keys κ and κ' statistically close.
- **Well-distributed logarithms:** For all generators $g, h \in \mathbb{G}$, all (κ', t) output by $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$ and all bit-strings $(X_i)_{1,\ldots,m}, (Z_i)_{1,\ldots,n} \in \Sigma^\ell$ such that $\forall i, j, X_i \neq Z_j$, we have $\Pr[a_{X_1} = \ldots, a_{X_m} = 0 \wedge a_{Z_1} \cdot \ldots \cdot a_{Z_n} \neq 0] \geq \delta$, where the probability is taken over the random coins used by $\mathsf{PHF.TrapGen}$ and $(a_{X_i}, b_{X_i}) := \mathsf{PHF.TrapEval}(t, X_i)$ and $(a_{Z_i}, b_{Z_i}) := \mathsf{PHF.TrapEval}(t, Z_i)$.

6.2 Instantiation with Waters Function

Let us consider the Waters function presented in [28].

Definition 16 (Multi-Generator PHF). Let $G = (\mathbb{G}_\lambda)$ be a group family, and $\ell = \ell(\lambda)$ a polynomial. We define $\mathcal{F} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ as the following group hash function:

- $\mathsf{PHF.Gen}(1^\lambda)$ outputs $\kappa = (h_0, \ldots, h_\ell) \xleftarrow{\$} \mathbb{G}^{\ell+1}$;
- $\mathsf{PHF.Eval}(\kappa, X)$ parses κ and $X = (x_1, \ldots, x_\ell) \in \{0,1\}^\ell$ and then outputs $\mathcal{F}_\kappa(X) = h_0 \prod_{i=1}^{\ell} h_i^{x_i}$.

This function was shown to be $(1, q, \delta)$-programmable with $\delta = O(1/(q\sqrt{\ell}))$ and $(2, 1, \delta)$-programmable with $\delta = O(1/\ell)$ (*cf.* [21]). However, this definition requires to generate and store $\ell+1$ group generators where ℓ is the bit-length of the messages one wants to hash. We consider a more general case where instead of hashing bit-per-bit we hash blocks of bits.

Definition 17 (Improved Multi-Generator PHF). Let $G = (\mathbb{G}_\lambda)$ be a group family, $\Sigma = \{0, \ldots, \tau\}$ a finite alphabet and $\ell = \ell(\lambda)$ a polynomial. We define $\mathcal{F} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$ as the following group hash function:

- $\mathsf{PHF.Gen}(1^\lambda)$ returns $\kappa = (h_0, \ldots, h_\ell) \xleftarrow{\$} \mathbb{G}^{\ell+1}$;
- $\mathsf{PHF.Eval}(\kappa, X)$ parses κ and $X = (x_1, \ldots, x_\ell) \in \Sigma^\ell$ and then outputs ${\mathcal{F}^+}_\kappa(X) = h_0 \prod_{i=1}^{\ell} h_i^{x_i}$.

Using a larger alphabet allows to hash from a larger domain with a smaller hash key, but it comes at a price: we show that the function is no longer $(2,1)$-programmable (*i.e.*, no longer $(2, 1, \delta)$ programmable for a non-negligible δ).

Theorem 18 ((2,1)-Programmability). *For any group family G with known order and $\tau > 1$, the function $\mathcal{F}^+$ is not a (2,1)-programmable hash function if the discrete logarithm problem is hard in G.*

Proof. Consider a discrete logarithm challenge (g, h) in a group $\mathbb{G}_\lambda$ and suppose by contradiction that the function $\mathcal{F}^+$ is $(2,1)$-programmable with $\tau \geq 2$ (*i.e.*, we suppose that there exist two probabilistic polynomial-time algorithms $(\mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval})$ satisfying Definition 15 for a non-negligible δ).

For any hash key κ' and trapdoor t generated by $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$, we can consider the messages $X_1 = (2,0), X_2 = (1,1), Z = (0,2)$.

With non-negligible probability over the random coins used by $\mathsf{PHF.TrapGen}$ we have $a_{X_1} = a_{X_2} = 0$ and $a_Z \neq 0$ where $(a_{X_1}, b_{X_1}) := \mathsf{PHF.TrapEval}(t, X_1)$, $(a_{X_2}, b_{X_2}) := \mathsf{PHF.TrapEval}(t, X_2)$ and $(a_Z, b_Z) := \mathsf{PHF.TrapEval}(t, Z)$. By the correctness property, we have

$$g^{a_Z} h^{b_Z} = \mathcal{F}(Z) = h_0 h_2^2 = (\mathcal{F}(X_2))^2 / \mathcal{F}(X_1) = h^{2b_{X_2}} / h^{b_{X_1}}$$

and we can extract the discrete logarithm of g in base h as follows:

$$\log_h(g) = \frac{2b_{X_2} - b_{X_1} - b_Z}{a_Z} \mod |\mathbb{G}_\lambda| \ .$$

However we still have the following interesting property:

Theorem 19 ((1,poly)-Programmability). *For any polynomial q and a group family G with groups of known order, the function $\mathcal{F}^+$ is a $(1, q, \delta)$-programmable hash function with a $\delta = \Omega(1/\tau q\sqrt{\ell})$.*

Remark 20. This theorem improves the result presented by Naccache in [23] where the lower bound on the $(1, q, \delta)$-programmability was only $\delta = \Omega(1/\tau q \ell)$.

Remark 21. In order to be able to sign all messages in a set $\mathcal{M}$, we have to consider parameters τ and ℓ such that $\tau^\ell \geq \#\mathcal{M}$, but the security is proved only if the value δ is non-negligible (*i.e.* if $\ell = \lambda^{O(1)}$ and $\tau = \lambda^{O(1)}$). In particular if $\mathcal{M}$ is of polynomial size in λ (which is the case in our WSN application with data aggregation), one can use $\tau = \#\mathcal{M}$ and $\ell = 1$ (namely, the Boneh-Boyen hash function [7]), and therefore get data confidentiality.

Proof. Let us first introduce some notation. Let $n \in \mathbb{N}^*$; for $j \in \{1, \ldots, n\}$, let A_j be independent and uniform random variables in $\{-1, 0, 1\}$. If we denote $2\sigma_j^2$ their quadratic moment, we have $2\sigma_j^2 = 2/3$ and $\sigma_j = \sqrt{1/3}$. We note $s_n^2 = \sum_{j=1}^{n} \sigma_j^2 = n/3$.

The Local Central Limit Theorem. Our analysis relies on a classical result on random walks, called the *Local Central Limit Theorem*[1] . It basically provides an approximation of $\Pr[\sum A_j = a]$ for independent random variables A_j. This is a version of the Central Limit Theorem in which the conclusion is strengthened from convergence of the law to locally uniform pointwise convergence of the densities. It is worded as follows in [14, *Theorem 1.1*], where ϕ and Φ are the standard normal density and distribution functions:

Theorem 22. *Let A_j be independent, integer-valued random variables where A_j has probability mass function f_j (for $j \in \mathbb{N}^*$). For each $j, n \in \mathbb{N}^*$, let $q(f_j) = \sum_k \min(f_j(k), f_j(k+1))$ and $Q_n = \sum_{j=1}^n q(f_j)$. Denote $S_n = A_1 + \cdots + A_n$. Suppose that there are sequences of numbers $(\alpha_n), (\beta_n)$ such that*

1. $\lim_{n\to\infty} \Pr[((S_n - \alpha_n)/\beta_n) < t] = \Phi(t), -\infty < t < \infty$,
2. $\beta_n \to \infty$,
3. *and* $\limsup \beta_n^2/Q_n < \infty$,

then[2] $\sup_k |\beta_n \Pr[S_n = k] - \phi((k - \alpha_n)/\beta_n)| \to 0$ *as* $n \to \infty$.

While those notations may seem a little overwhelming, this can be easily explained in our case. With $A_j \in \{-1, 0, 1\}$ with probability 1/3 for each value.

1. It requires the variables to verify the Lindeberg-Feller theorem. However as long as the variables verify Lindeberg's condition[3], this is true for $\beta_n = s_n$ and $\alpha_n = 0$.
2. In our application, $\beta_n = s_n = \sqrt{n/3}$, so again we comply with the condition.
3. Since $f_j(k)$ is simply the probability that A_j equals k, then $q(f_j) = 2/3$. This leads to $Q_n = 2n/3$. As a consequence, $\beta_n^2/Q_n = 1/2$.

So we have: $\sup_k |\beta_n \Pr[S_n = k] - \phi((k - \alpha_n)/\beta_n)| \to 0$, that is, in our case

$$\sup_k |\sqrt{n/3} \Pr[S_n = k] - \phi(k/\sqrt{n/3})| \to 0 \ .$$

We solely focus on the case $k = 0$: since $\phi(0) = 1/\sqrt{2\pi}$, $\Pr[S_n = 0] = \Theta(1/\sqrt{n})$. In addition, it is clear that $\Pr[S_n = k] \leq \Pr[S_n = 0]$ for any $k \neq 0$ (*cf.* [21]).

Lemma 23. *Let $(A_{ij})_{[\![1,n]\!]\times[\![1,J]\!]}$ be independent, integer-valued random variables in $\{-1, 0, 1\}$, then $\forall X \in [\![1, \tau]\!]^n$, $\Pr[\sum_{i=1}^n \sum_{j=1}^J X_i A_{ij} = 0] = \Omega(1/\tau\sqrt{nJ})$, where the probability distribution is over the A_{ij}.*

This lemma will be useful to prove the lower bound in the following, we only consider word with no null coefficient X_i, if a X_i is null, we simply work with a shorter random walk of length $J \cdot (n-1)$ instead of Jn.

Proof. Let us denote d_{ij}, the random variable defined as $X_i A_{ij}$: they are independent, integer-valued random variables. As above, $s_n^2 = \sum_{i=1}^n \sum_{j=1}^J \sigma_j^2 = \sum_{i=1}^n JX_i^2/3$. So $nJ/3 \leq s_n^2 \leq n\tau^2 J/3$.

1. Lindeberg's condition is verified. As soon as $n > 3\tau/J\epsilon^2$ we have $\epsilon s_n > \tau$ and so $|d_{ij}| < s_n$, and so once again the sum is null.
2. $s_n \to \infty$.
3. Each $d_{ij} \in \{-X_i, 0, X_i\}$ with probability 1/3 for each value, so $q(f_{ij}) = 2/3$ and $Q_n = \sum_{i,j} q(f_{ij}) = 2nJ/3$. So $\beta_n^2/Q_n \leq (n\tau J/3)/(2nJ/3) \leq \tau/2 < \infty$.

Then we can apply the Local Central Limit Theorem to the d_{ij}'s, and conclude: $\Pr[\sum_{i=1}^n \sum_{j=1}^J X_i A_{ij} = 0] = \Theta(1/s_n) = \Theta(1/\tau\sqrt{nJ})$.

[1] The main idea here is to show that even if the probability that the studied random walks end in 0 is maximal, this probability in neither negligible nor overwhelming.

[2] The so-called Berry-Esseen theorem gives the rate of convergence of this supremum.

[3] Lindeberg's condition is a sufficient criteria of the Lindeberg-Feller theorem, for variables with a null expected value it requires that $\forall \epsilon > 0, \lim_{n\to\infty} 1/s_n^2 \sum_{j=1}^n E[A_j^2 \cdot 1_{\{|A_j| > \epsilon s_n\}}] \to 0$. In our case, as soon as $n > 3/\epsilon^2$, we have $|A_j| \leq 1 \leq \epsilon\sqrt{n/3} \leq \epsilon s_n$, so the sum is zero. ($1_{\{|A_j|>\epsilon s_n\}}$ is the indicator function of variables greater that ϵs_n)

In the following, we will denote $a(X) = \sum_{i=1}^{n} a_i X_i$, where $X \in \{0, \ldots, \tau\}^n$. The probabilities will be over the a_{ij}'s variables while X and Y are assumed to be chosen by the adversary. Our goal is to show that even for bad choices of X and Y, a random draw of a_{ij}'s provides enough freedom.

Let $J = J(\lambda)$ be a positive function. We define the following two probabilistic polynomial-time algorithms ($\mathsf{PHF.TrapGen}, \mathsf{PHF.TrapEval}$):

- $\mathsf{PHF.TrapGen}(1^\lambda, g, h)$**:** which chooses some independent and uniform elements $(a_{ij})_{(0,\ldots,\ell),(1,\ldots,J)}$ in $\{-1, 0, 1\}$, and random exponents $(b_i)_{(0,\ldots,\ell)}$. It sets $a_i = \sum_{j=1}^{J} a_{ij}$ and $h_i = g^{a_i} h^{b_i}$ for $i \in \{0, \ldots, \ell\}$. It then outputs the hash key $\kappa = (h_0, \ldots, h_\ell)$ and the trapdoor $t = (a_0, b_0, \ldots, a_\ell, b_\ell)$.
- $\mathsf{PHF.TrapEval}(t, X)$**:** which parses $X = (X_1, \ldots, X_\ell) \in \Sigma^\ell = \{0, \ldots, \tau\}^\ell$ and outputs $a_X = a_0 + \sum a_i X_i$ and $b_X = b_0 + \sum b_i X_i$.

As this definition verifies readily the syntactic and correctness requirements, we only have to prove the two other ones. We stress the importance of the hardwired 1 in front of a_0 this allows us to consider multisets $X' = 1 :: X$ and $Y' = 1 :: Y$, and so there is no k such that $X' = kY'$. And we also stress that $a_i = \sum_{j=1}^{J} a_{ij}$ is already a random walk of length J (described by the a_{ij}), on which we can apply the Local Central Limit Theorem and so $\Pr[a_i = 0] = \Theta(1/\sqrt{J})$. By noticing that summing independent random walks is equivalent to a longer one and applying the Local Central Limit Theorem, we have:

$$\Theta(1/\tau\sqrt{(\ell+1)J}) \leq \Pr[a(X') = 0] \leq \Theta(1/\sqrt{J}) \ .$$

To explain further the two bounds:

- For the upper bound: we consider X fixed, and note $t = \sum_{i=1}^{\ell} a_i X_i$, by construction a_i are independent, so a_0 is independent from t then
$$\Pr[a(X') = 0] = \Pr[a_0 = -t] \leq \Pr[a_0 = 0] \leq \Theta(1/\sqrt{J})$$
using the above remark that a random walk is more likely to reach 0 than any other value, and a_0 is a random walk of length J.
- For the lower bound, we proceed by recurrence on ℓ, to show
$$H_\ell : \Theta(1/\tau\sqrt{(\ell+1)J}) \leq \Pr[a(X') = 0] \quad (\text{where } X' \in 1 :: [\![0, \tau]\!]^\ell).$$
For $\ell = 0$, we consider $X' = 1$, we have a random walk of length J, so $\Theta(1/\tau\sqrt{J}) \leq \Theta(1/\sqrt{J}) \leq \Pr[a(X') = 0]$. We note $X_0 = 1$ for the hardwired 1 in X'. Let us suppose the property true at rank k, let us prove it at rank $k+1$:
  - If $\exists i_0, X_{i_0} = 0$ then we can consider a random walk of length k and apply the previous step, and conclude because $\Theta(1/\tau\sqrt{(k+1)J}) \leq \Theta(1/\tau\sqrt{kJ})$
  - Else, one can apply Lemma 23 to conclude.

  Therefore, $\forall \ell, \forall X' \in 1 :: [\![0, \tau]\!]^\ell, \Theta(1/\tau\sqrt{(\ell+1)J}) \leq \Pr[a(X') = 0]$.

We can now deduce that $\forall X, Y \in [\![0, \tau]\!]^\ell$ with $X \neq Y$: $\Pr[a(Y') = 0 | a(X') = 0] \leq \Theta(1/\sqrt{J})$. This can easily be seen by noting i_0 the first index where $Y_i \neq X_i$. We will note $\bar{X'} = X' - X_{i_0}$, in the following we will use the fact that $a(X') = 0 \Leftrightarrow a(\bar{X'}) = -a_{i_0} X_{i_0}$.[4]

$$\begin{aligned}
\Pr[a(Y') = 0 | a(X') = 0] &\leq \Pr[a(Y') = a(X') | a(X') = 0] \\
&\leq \Pr[Y_{i_0} a_{i_0} + a(\bar{Y'}) = X_{i_0} a_{i_0} + a(\bar{X'}) | a(X') = 0] \\
&\leq \max_t \Pr[(Y_{i_0} - X_{i_0}) a_{i_0} = t | a(\bar{X'}) = -X_{i_0} a_{i_0}] \quad (1) \\
&\leq \max_{s,t'} \Pr[a_{i_0} = t' | a(\bar{X'}) = s] \quad (2) \\
&\leq \max_{t'} \Pr[a_{i_0} = t'] \quad (3) \\
&\leq \Pr[a_{i_0} = 0] \leq \Theta(1/\sqrt{J})
\end{aligned}$$

(1) We start with $(Y_{i_0} - X_{i_0}) a_{i_0} = a(\bar{X'}) - a(\bar{Y'})$, and then consider the maximum probability for all values $a(\bar{X'}) - a(\bar{Y'})$.

[4] $X \neq Y$ so i_0 exists, and thanks to the hardwired 1 we do not have to worry about Y' being a multiple of X'

(2) We consider the maximum probability for all values of $-X_{i_0}a_{i_0}$.
(3) a_{i_0} and $a(\bar{X}')$ are independent.

Hence, for all $X_1, Y_1, \ldots, Y_q$, we have

$$\begin{aligned}\Pr[a_{X_1} = 0 \wedge a_{Y_1}, \ldots, a_{Y_q} \neq 0] &= \Pr[a_{X_1} = 0] \Pr[a_{Y_1}, \ldots, a_{Y_q} \neq 0 | a_{X_1} = 0] \\ &\geq \Theta(1/\tau\sqrt{\ell J}) \left(1 - \sum_{i=1}^{q} \Pr[a_{Y_i} = 0 | a_{X_1} = 0]\right) \\ &\geq \Theta(1/\tau\sqrt{(\ell+1)J})(1 - q\Theta(1/\sqrt{J})) \ .\end{aligned}$$

Now we set $J = q^2$, to obtain the result. In that case the experiment success is lower-bounded by something linear in $1/(q\tau\sqrt{\ell+1})$.

Studying the programmability of such functions is important. Having a $(q,1)$-programmable hash is a sufficient condition to instantiate a BLS-like signature scheme. Our result on the non-$(2,1)$ programmability over a non-binary alphabet while non-discarding the possibility says that it is probably not a good idea to try to instantiate such a scheme using Waters PHF. The $(1,q)$-programmability says that the Programmable Hash Function can be used in a signature scheme / IBE scheme, where we need to simulate q queries and use one challenge, this paper proposes a construction of such signature scheme and presents various applications.

References

1. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In CRYPTO 2010, volume 6223 of LNCS, pages 209–236. Springer, August 2010.
2. Masayuki Abe and Eiichiro Fujisaki. How to date blind signatures. In ASIACRYPT 1996, volume 1163 of LNCS, pages 244–251. Springer, November 1996.
3. Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In CRYPTO 2000, volume 1880 of LNCS, pages 271–286. Springer, August 2000.
4. Josh Cohen Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *26th Annual ACM Symposium on Theory of Computing*, pages 544–553. ACM Press, May 1994.
5. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In PKC 2011, volume 6571 of LNCS. pages 403–422, Springer, May 2010.
6. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In TCC 2012, volume 7194 of LNCS, pages 94–111, Springer, March 2012.
7. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity-based encryption without random oracles. In EUROCRYPT 2004, volume 3027 of LNCS, pages 223–238. Springer, May 2004.
8. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In CRYPTO 2004, volume 3152 of LNCS, pages 41–55. Springer, August 2004.
9. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In EUROCRYPT 2006, volume 4004 of LNCS, page 427–444. Springer, May / June 2006.
10. David Chaum. Blind signatures for untraceable payments. In CRYPTO 1982, pages 199–203. Plenum Press, New York, USA, 1983.
11. Benoît Chevallier-Mames, Pierre-Alain Fouque, Julien P. Stern, David Pointcheval, and Jacques Traoré. On Some Incompatible Properties of Voting Schemes. In *Actes du IAVoSS Workshop On Trustworthy Elections (WOTE 2006)*, Cambridge, UK, 2006.
12. Stéphanie Delaune and Steve Kremer. Formalising security properties in electronic voting protocols. Deliverable AVOTE 1.2, (ANR-07-SESU-002), April 2010. 17 pages.
13. Ivan Damgård and Jesper Buus Nielsen Perfect Hiding and Perfect Binding Universally Composable Commitment Schemes with Constant Expansion Factor. In Crypto 2002, volume 2442 of LNCS, page 581–596. Springer, August 2002.
14. Burgess Davis and David McDonald. An elementary proof of the local central limit theorem. *Journal of Theoretical Probability*, 8(3), pages 693–701, jul 1995.
15. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In CRYPTO 2006, volume 4117 of LNCS, pages 60–77. Springer, August 2006.
16. Georg Fuchsbauer. Commuting signatures and verifiable encryption. In EUROCRYPT 2011, volume 6632 of LNCS, pages 224–245. Springer, May 2011.
17. Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In CRYPTO 2011, volume 6841 of LNCS, pages 630–648. Springer, August 2011.
18. Shafi Goldwasser and Silvio Micali, Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299,1984.
19. 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, Apr. 1988.

20. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In EUROCRYPT 2008, volume 4965 of LNCS, pages 415–432. Springer, April 2008.
21. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In CRYPTO 2008, volume 5157 of LNCS, pages 21–38. Springer, August 2008.
22. Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In EUROCRYPT 2006, volume 4004 of LNCS, pages 465–485. Springer, 2006.
23. David Naccache. Secure and *practical* identity-based encryption. Cryptology ePrint Archive, Report 2005/369, 2005.
24. Tatsuaki Okamoto. Efficient blind and partially blind signatures without random oracles. In TCC 2006, volume 3876 of LNCS, pages 80–99. Springer, March 2006.
25. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.
26. Dominique Schröder and Dominique Unruh. Security of Blind Signatures Revisited, In PKC 2012, volume 7293 of LNCS, pages 662–679. Springer, May 2012.
27. Jae Hong Seo and Jung Hee Cheon. Beyond the limitation of prime-order bilinear groups, and round optimal blind signatures. In TCC 2012, volume 7194 of LNCS, pages 133–150. Springer, 2012.
28. Brent R. Waters. Efficient identity-based encryption without random oracles. In EUROCRYPT 2005, volume 3494 of LNCS, pages 114–127. Springer, May 2005.

A Asymmetric Version

All the schemes presented so far can be adapted for asymmetric groups. The main, and only difference, comes from the Groth-Sahai commitments. As symmetric bilinear groups are in general less efficient than *asymmetric* groups, we show how to instantiate our primitive with Groth-Sahai commitments in an asymmetric pairing-friendly group setting, relying on the SXDH assumption.

A.1 Assumptions

The security of Waters signatures in asymmetric bilinear groups was proven in [5] under the following variant of the CDH assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$.

Definition 24 (Advanced Computational Diffie-Hellman problem (CDH^+)). Let $(\mathbb{G}_1, \mathbb{G}_2)$ be groups of prime order p with (g_1, g_2) as respective generators and e an admissible bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

ElGamal encryption is secure under the DDH assumption, which should hold in both $\mathbb{G}_1$ and $\mathbb{G}_2$ for a more efficient variant of Groth-Sahai proofs to be secure.

Definition 25 (Decisional Diffie-Hellman Assumption (DDH)). Let $\mathbb{G}$ be a cyclic group of prime order p. The DDH assumption states that given a 4-tuple $(g, g^a, g^b, g^c) \in \mathbb{G}$, it is hard to determine whether $c = ab$.

Definition 26 (Symmetric external Diffie-Hellman Assumption (SXDH) [8]). Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. The SXDH assumption states that DDH holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

A.2 Groth-Sahai Commitments

We will use SXDH-based Groth-Sahai commitments, which are a direct transposition of the previous ones in an asymmetric setting and replace double linear encryption by a double ElGamal encryption in a pairing friendly environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible bilinear map, for three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, of prime order p, generated by g_1, g_2 and $g_t = e(g_1, g_2)$ respectively.

The commitment key consists of $\mathbf{u}_1 = (u_{1,1}, u_{1,2}), \mathbf{u}_2 = (u_{2,1}, u_{2,2}) \in \mathbb{G}_1^2$ and $\mathbf{v}_1 = (v_{1,1}, v_{1,2}), \mathbf{v}_2 = (v_{2,1}, v_{2,2}) \in \mathbb{G}_2^2$. We write

$$\mathbf{u} = \begin{pmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{pmatrix} = \begin{pmatrix} u_{1,1}\ u_{1,2} \\ u_{2,1}\ u_{2,2} \end{pmatrix} \quad \text{and} \quad \mathbf{v} = \begin{pmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \end{pmatrix} = \begin{pmatrix} v_{1,1}\ v_{1,2} \\ v_{2,1}\ v_{2,2} \end{pmatrix}.$$

– Binding initialization of the parameters is: $\boldsymbol{u_1} = (g_1, u)$ with $u = g_1^{\lambda}$ and $\boldsymbol{u_2} = \boldsymbol{u_1}^{\mu}$ with $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, which means that $\mathbf{u}$ is a Diffie-Hellman tuple in $\mathbb{G}_1$, since $\boldsymbol{u_1} = (g_1, g_1^{\lambda})$ and $\boldsymbol{u_2} = (g_1^{\mu}, g_1^{\lambda\mu})$.
– Hiding initialization: we will use instead $\boldsymbol{u}_2 = \boldsymbol{u_1}^{\mu} \odot (1, g_1)^{-1}$: $\boldsymbol{u_1} = (g_1, g_1^{\lambda})$ and $\boldsymbol{u_2} = (g_1^{\mu}, g_1^{\lambda\mu-1})$, and analogously for in $\mathbb{G}_2$ for $\boldsymbol{v}$.

Under the SXDH assumption, the two initializations are indistinguishable.

Commitments to Group Elements. To commit to $X \in \mathbb{G}_1$, one chooses randomness $s_1, s_2 \in \mathbb{Z}_p$ and sets

$$\mathcal{C}(X) = (1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} = (u_{1,1}^{s_1} \cdot u_{2,1}^{s_2}, X \cdot u_{1,2}^{s_1} \cdot u_{2,2}^{s_2}) \ .$$

A simulator that knows the discrete logarithm λ of u in basis g_1 can extract X in the perfectly binding setting. The commitment in $\mathbb{G}_2$ follows the same rules, with $\boldsymbol{v}$ and g_2 instead of $\boldsymbol{u}$ and g_1.

Commitments to Scalars. One actually commits to g_1^x, from which x can be extracted if this is a bit.

Proofs. This time, a Groth-Sahai proof is a pair of elements $(\pi, \theta) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. One has to pay attention to the fact that Groth-Sahai bit-by-bit proofs in SXDH require bits to be committed both in $\mathbb{G}_1$ and $\mathbb{G}_2$ and thus require to use 2 quadratic equations by bit.

A.3 Partially Blind Signatures with Perfect Blindness

The construction is completely straightforward. If we follow the steps from the DLin-version: We will need 2 group elements for the commitment to M in $\mathbb{G}_1$, 4 group elements to commit Y_1, Y_2 in $\mathbb{G}_1$, the proofs will require 4 group elements in $\mathbb{G}_2$. We will need 6ℓ elements in each group to commit to M and prove we indeed committed it bit-by-bit, and 2 extra group elements in $\mathbb{G}_2$ to prove c_2 is well-formed. The signatures on the committed elements will require 3 groups elements in $\mathbb{G}_1$ and one in $\mathbb{G}_2$. Therefore the overall scheme will require $(6\ell + 9, 6\ell + 7)$ group elements communication.