Signatures on Randomizable Ciphertexts

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

[1] École normale supérieure - CNRS - INRIA, Paris, France
[2] Dept. Computer Science, University of Bristol, UK

Abstract. Randomizable encryption allows anyone to transform a ciphertext into a fresh ciphertext of the same message. Analogously, a randomizable signature can be transformed into a new signature on the same message. We combine randomizable encryption and signatures to a new primitive as follows: given a signature on a ciphertext, anyone, knowing neither the signing key nor the encrypted message, can randomize the ciphertext and *adapt* the signature to the fresh encryption, thus maintaining public verifiability. Moreover, given the decryption key and a signature on a ciphertext, one can compute ("extract") a signature on the encrypted plaintext. As adapting a signature to a randomized encryption contradicts the standard notion of unforgeability, we introduce a weaker notion stating that no adversary can, after querying signatures on ciphertexts of its choice, output a signature on an encryption of a *new* message. This is reasonable since, due to extractability, a signature on an encrypted message can be interpreted as an encrypted signature on the message.
Using Groth-Sahai proofs and Waters signatures, we give several instantiations of our primitive and prove them secure under classical assumptions in the standard model and the CRS setting. As an application, we show how to construct an efficient non-interactive receipt-free universally verifiable e-voting scheme. In such a scheme a voter cannot prove what his vote was, which precludes vote selling. Besides, our primitive also yields an efficient round-optimal blind signature scheme based on standard assumptions, and namely for the classical Waters signature.

1 Introduction

Homomorphic cryptographic primitives have already found numerous applications. A nice side effect of homomorphic encryption is that ciphertexts can be *randomized*: given a ciphertext, anyone can—without knowing the encrypted message—produce a fresh ciphertext of the same message. E-voting schemes make use of homomorphic encryption: users encrypt their votes under such a scheme (and add proofs and signatures), so combining the ciphertexts leads to an encryption of the election result. All signed encryptions are then made public and *verifiable*, enabling the users to check that their vote was counted, and anybody to verify the correctness of the final tally. Now, if instead of directly using a user's ciphertext, the voting center first randomizes it and proves that it did so correctly, in a non-transferable way, then users are prevented from proving the content of their vote by opening it. This deters from *vote selling*, since someone buying a vote has no means to check whether the user voted as told.

However, such a (non-transferable) proof of correct randomization is costly, and the randomization breaks most of the proofs of validity of the individual ciphertexts and signatures, and thus universal verifiability. More efficient techniques are thus desirable, and this is precisely the motivation for our new primitive: it allows to adapt signatures and proofs on the content of a ciphertext when the ciphertext is randomized, that is, when the content itself is not modified. This makes proofs of correct randomization obsolete, since after receiving an encrypted vote with a validity proof and a signature from a user, the voting center can randomize the ciphertext as well as the proof and signature accordingly, which preserves universal verifiability. However, because of the randomization of the encryption of the vote, the voter is not able to open nor link this encrypted ballot to anything: no receipt can be built.

In contrast to e-voting, there are situations where encryption and signing are not performed by the same person; consider a user that encrypts a message and asks for a signature on the ciphertext. Assume now that the user can compute from this an actual signature on the message (rather than on an encryption thereof). The signature on the ciphertext could then be seen as an encrypted signature on the message,

⋆ Work done while at École normale supérieure, Paris, France.

which can be decrypted by the user. This resembles a blind signature, as the signer made a signature on an unknown message; but not quite, since he may later recognize the signature (knowing the random coins he used) and thus break blindness. A possible remedy are *randomizable signatures*, which allow to transform a given signature into a new one on the same message. Such signatures, a classical example being Waters signatures [Wat05], do not satisfy *strong* unforgeability, which requires that it be impossible even to create a new signature on a signed message. As we show, this apparent weakness is actually a feature, as it can be exploited to achieve *unlinkability*: the blindness property is achieved by randomizing a signature after reception.

Fischlin [Fis06] gives a generic construction of *round-optimal* blind signatures which has been efficiently instantiated recently [Fuc09, AFG+10]. To prevent the signer from linking a blind signature to the signing session, they define a blind signature as a (non-interactive) *proof of knowledge* of a signature. This makes blind signatures significantly longer than signatures of the underlying scheme, which can be avoided using randomizable signatures. In Fischlin's scheme a blind signature is a proof of knowledge of a signature on a ciphertext together with a proof that the ciphertext decrypts to the message. In the scheme in [Fuc09], the user obtains an actual signature on the message, of which he proves knowledge. We go one step further: again, the user can extract a signature on the message; but instead of making a proof of knowledge, it suffices to simply randomize it to make it unlinkable. A blind signature has therefore the same format as the underlying signatures and, in addition to being round-optimal, is thus short.

Getting back to the receipt-free voting schemes, unlinkability of ciphertexts after randomization is guaranteed by the semantic security of the encryption scheme. If at the same time of randomizing the ciphertext, the voting center adapts the proofs (validity of the ciphertext and signature by the voter), the ciphertexts remain unlinkable, under the conditions that they are valid ciphertext and signed by a given voter. As a consequence, two valid encrypted votes signed by the same voter are unlinkable: there is no way to know nor prove (even for the voter) if they contain the same vote or any specific vote, which prevents the voter from selling his vote.

Our contribution. We first introduce the notion of *signatures on randomizable ciphertexts*: given a signature on a ciphertext, anyone, knowing neither the signing key nor the encrypted message, can randomize the ciphertext and *adapt* the signature to the fresh encryption. A pair of a ciphertext and a signature on it can thus be randomized simultaneously and consistently.

Since adapting a signature on one ciphertext to a signature on another ciphertext contradicts the standard notion of unforgeability for signatures, we define a weaker notion, which still implies the security of our applications: unforgeability of signatures on randomizable ciphertexts means that the only thing an adversary can do is produce signatures on encryptions of messages of which he already knows a signature on an encryption; but he cannot make a signature on an encryption of a *new* message. Formally, no adversary can, after querying signatures on ciphertexts of its choice, output a signature on a ciphertext whose decryption is different from the decryptions of all queried ciphertexts.

We then extend our primitive to *extractable* signatures on randomizable ciphertexts: given the decryption key, from a signature on a ciphertext one can *extract* a signature on the encrypted plaintext. This enables the user in a blind-signature scheme to recover a signature on the message after the signer has signed an encryption of it.

Instantiations. We give several instantiations of extractable signatures on randomizable ciphertexts, all of which are based on weak assumptions. Our constructions use the following building blocks, from which they inherit their security: Witness-indistinguishable Groth-Sahai proofs for languages over pairing-friendly groups [GS08] and Waters signatures derived from the scheme in [Wat05] and used in [BW06]. Since verification of Waters signatures is a statement of the language for Groth-Sahai proofs, these two building blocks combine smoothly. The first instantiation of our new primitive is in symmetric pairing-friendly elliptic curves and additionally uses linear encryption [BBS04]. Both unforgeability and semantic security of this construction

rely solely on the decision linear assumption (DLin). Due to space limitations, an instantiation with improved efficiency, in *asymmetric* bilinear groups, using ElGamal encryption and the SXDH variant of Groth-Sahai proofs is postponed to Appendices C and D. This setting requires to transfer Waters' signature scheme to asymmetric groups. Whereas standard Waters signatures are secure under the computational Diffie-Hellman assumption (CDH), we prove our variant secure under a slightly stronger assumption, we term CDH^+, where some additional elements in the second group are given to the adversary. The following table details the size of a ciphertext-signature pair, where the parameter k denotes the bit length of a message:

Symmetric Pairing	$\mathbb{G}$
Waters + Linear	$9k + 24$

Asymmetric Pairing	$\mathbb{G}_1$	$\mathbb{G}_2$
Waters + ElGamal	$6k + 7$	$6k + 5$

Applications. Using our new primitive, we immediately obtain a reasonably efficient round-optimal blind-signature scheme based on standard assumptions. Moreover, exploiting the fact that our encryption is homomorphic, we construct a non-interactive receipt-free universally verifiable e-voting scheme as follows: the user encrypts his vote, proves its validity, and sends the encryption, a signature on it, and the proof to the voting center. The latter can now randomize the ciphertext, *adapt* both the proof and the user's signature, and publish them. After the results are announced, the user can verify his signature, which convinces him that the randomized ciphertext still contains his original vote due to our notion of unforgeability; however he cannot prove to anyone what his vote was.

Related work. The issue of signing messages that are only available as an encryption was already addressed by Fuchsbauer in [Fuc10]. He introduced *commuting signatures and verifiable encryption* where, given a ciphertext, a signer can produce a verifiably encrypted signature on the plaintext. These encrypted signatures can be randomized and used to construct the first delegatable anonymous credentials [BCC+09] with a non-interactive delegation protocol.

We avoid (randomizable) verifiable encryption of signatures by using signatures that are themselves randomizable. In our instantiation of round-optimal blind signatures, the blind signature is an *actual* signature rather than a verifiable encryption of it. Moreover, our construction is based on standard assumptions, whereas [Fuc10] relies on a "q-type" assumption. The efficiency of the two approaches is comparable when signing short messages, as required by our application to e-voting—since votes typically consist of only a few bits. We note however that the size of our ciphertexts is linear in the bit length of the message.

Organization. In the next section, we present the primitive and the security model. We then give two instantiations in symmetric bilinear groups based on the decision linear assumption. We first fix ideas using standard Waters signatures and then define a variant which yields a significant efficiency improvement of our instantiation, proven secure under the same assumptions. Due to space limitations, our instantiation in asymmetric groups based on ElGamal encryption and an asymmetric variant of Waters signatures is deferred to the Appendices C and D. In the last section, we illustrate applications of our primitive.

2 Definitions

This section presents the global framework and the security model for our new concept of signatures on ciphertexts (or commitments). We thus first briefly recall the basics of signatures and encryption. We then combine both into a single scheme.

2.1 Notations for Signature and Encryption

Definition 1 (Encryption Scheme). $\mathcal{E} = (\mathsf{Setup}, \mathsf{EKeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$*:*

– $\mathsf{Setup}(1^k)$*, where k is the security parameter, generates the global parameters* param *of the scheme;*

– $\mathsf{EKeyGen}(\mathsf{param})$ *generates a pair of keys, the public (encryption) key* pk *and the associated private (decryption) key* dk*;*
– $\mathsf{Encrypt}(\mathsf{pk}, m; r)$ *produces a ciphertext* c *on the input message* $m \in \mathcal{M}$ *and the public key* pk*, using the random coins* $r \in \mathcal{R}_e$*;*
– $\mathsf{Decrypt}(\mathsf{dk}, c)$ *decrypts the ciphertext* c *under the private key* dk*; it outputs the plaintext, or* $\perp$ *if the ciphertext is invalid.*

Definition 2 (Signature Scheme). $\mathcal{S} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verif})$*:*

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

In Waters' signature scheme, the signing algorithm first transforms the message to $F = \mathcal{F}(M)$, where $\mathcal{F}$ is a hash function. Given F, the value of M is not required for signing and verification, but for the security guarantee. We could thus replace M by the pair (F, Π_M), where Π_M is a proof of knowledge of a preimage of F under the function $\mathcal{F}$ (which we assume implicitly). We define $\mathsf{Sign}(\mathsf{sk}, (F, \Pi_M); s)$ and $\mathsf{Verif}(\mathsf{vk}, (F, \Pi_M), \sigma)$ that extend the above definitions.

2.2 Signatures on Ciphertexts

We now define a scheme of signatures on ciphertexts. Note that this definition can be adapted for commitments, when one uses a perfectly binding commitment scheme, which uniquely defines the committed input.

Definition 3 (Signatures on Ciphertexts). $\mathcal{SC} = (\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{EKeyGen}, \mathsf{Encrypt}, \mathsf{Sign}, \mathsf{Decrypt}, \mathsf{Verif})$ *is defined as follows:*

– $\mathsf{Setup}(1^k)$*, where* k *is the security parameter, generates the global parameters* param_e *and* param_s *for the associated encryption and signature schemes;*
– $\mathsf{EKeyGen}(\mathsf{param}_e)$ *generates a pair of keys, the encryption key* pk *and the associated decryption key* dk*;*
– $\mathsf{SKeyGen}(\mathsf{param}_s)$ *generates a pair of keys, the verification key* vk *and the signing key* sk*;*
– $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, m, r)$ *produces a ciphertext* c *on input the message* $m \in \mathcal{M}$ *and the encryption key* pk*, using the random coins* $r \in \mathcal{R}_e$*. This ciphertext is intended to be later signed under the signing key associated to the verification key* vk *(the field for* vk *can be empty if the signing algorithm is universal and does not require a ciphertext specific to the signer);*
– $\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c; s)$*, on input a ciphertext* c *and a signing key* sk*, using the random coins* $s \in \mathcal{R}_s$*, produces a signature* σ*, or* $\perp$ *if the ciphertext* c *is not valid (w.r.t.* pk*, and possibly* vk *associated to* sk*);*
– $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{vk}, c)$ *decrypts the ciphertext* c *under the private key* dk*. It outputs the plaintext, or* $\perp$ *if* c *is invalid (w.r.t.* pk*, and possibly* vk*);*
– $\mathsf{Verif}(\mathsf{vk}, \mathsf{pk}, c, \sigma)$ *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 ciphertext* c*, with respect to* pk*, and possibly* vk*).*

Classical security notions could still be applied to this signature scheme, but we want ciphertexts and signatures to be efficiently malleable, as long as the plaintext is not affected. This will be useful for probabilistic schemes, and even more so for the randomizable scheme we will present below. In the classical definition of *existential unforgeability* (EUF) [GMR88], a new signature on an already signed message is not considered

$\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(k)$
$(\mathsf{param}_e, \mathsf{param}_s) \leftarrow \mathsf{Setup}(1^k)$; $\mathsf{SM} := \emptyset$
$\{(\mathsf{pk}_i, \mathsf{dk}_i)\} \leftarrow \mathsf{EKeyGen}(\mathsf{param}_e)$; $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param}_s)$
$(\mathsf{pk}_j, c, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}(\mathsf{sk},\cdot,\cdot)}(\mathsf{param}_s, \mathsf{param}_e, \mathsf{vk}, \{(\mathsf{pk}_i, \mathsf{dk}_i)\})$;
$m \leftarrow \mathsf{Decrypt}(\mathsf{dk}_j, \mathsf{vk}, c)$
`IF` $m = \perp$ `OR` $m \in \mathsf{SM}$ `OR` $\mathsf{Verif}(\mathsf{vk}, \mathsf{pk}_j, c, \sigma) = 0$ `RETURN` 0
`RETURN` 1

Fig. 1. Unforgeability of signatures on ciphertexts

a valid forgery—as opposed to strong unforgeability (SUF). When signing ciphertexts, EUF would consider a signature on a randomized ciphertext as a valid forgery. But if the ciphertext is equivalent to an already signed ciphertext (i.e. it encrypts the same plaintext), this may not be critical in some applications; in particular if we decrypt later anyway and a decrypted message-signature pair is unforgeable. We thus define the most appropriate unforgeability (UF) notion for signatures on ciphertexts:
$\mathcal{SC}$ is *unforgeable* if, for any polynomial-time adversary $\mathcal{A}$, the advantage $\mathsf{Succ}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(k) := \Pr[\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}(k) = 1]$ is negligible, with $\mathsf{Exp}^{\mathsf{uf}}_{\mathcal{SC},\mathcal{A}}$ defined in Figure 1. There, $\mathsf{Sign}(\mathsf{sk}, \cdot, \cdot)$ is an oracle that takes as input a previously generated encryption key pk_i and a ciphertext c, and generates a signature σ on it (if the ciphertext is valid). It also updates the set SM of signed plaintexts with $m = \mathsf{Decrypt}(\mathsf{dk}_i, \mathsf{vk}, c)$, if the latter exists.

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

2.3 Signatures on Randomizable Ciphertexts

Our primitive is based on an encryption scheme and a signature scheme. Since we want randomizability, we start by enhancing these schemes with randomization algorithms satisfying certain properties.

Definition 4 (Randomizable Encryption Scheme).
Let $(\mathsf{Setup}, \mathsf{EKeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *be an encryption scheme with the following additional algorithm:*

- $\mathsf{Random}(\mathsf{pk}, c; r')$ *produces a new ciphertext* c'*, equivalent to the input ciphertext* c*, under the public key* pk*, using the additional random coins* $r' \in \mathcal{R}_e$*.*

An encryption scheme is called randomizable *if for any* $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$*,* $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{EKeyGen}(\mathsf{param})$*, message* $m \in \mathcal{M}$*, coins* $r \in \mathcal{R}_e$*, and ciphertext* $c = \mathsf{Encrypt}(\mathsf{pk}, m; r)$*, the following distributions are statistically indistinguishable:* $\mathcal{D}_0 = \{r' \xleftarrow{\$} \mathcal{R}_e : \mathsf{Encrypt}(\mathsf{pk}, m; r')\}$ *and* $\mathcal{D}_1 = \{r' \xleftarrow{\$} \mathcal{R}_e : \mathsf{Random}(\mathsf{pk}, c; r')\}$*.*

Definition 5 (Randomizable Signature Scheme).
Let $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{Sign}, \mathsf{Verif})$ *be a signature scheme, with the following additional algorithm:*

- $\mathsf{Random}(\mathsf{vk}, (F, \Pi_M), \sigma; s')$ *produces a new signature* σ' *valid under* vk *from* σ *on a message* M *given as* $F = \mathcal{F}(M)$ *and a proof* Π_M *of knowledge of* M*, using the additional random coins* $s' \in \mathcal{R}_s$*.*

A signature scheme is called randomizable *if for any* $\mathsf{param} \leftarrow \mathsf{Setup}(1^k)$*,* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$*, message* $M \in \mathcal{M}$*, proof of knowledge* Π_M *of the preimage* M *of* $F = \mathcal{F}(M)$*, random* $s \in \mathcal{R}_s$*, signature* $\sigma = \mathsf{Sign}(\mathsf{sk}, (F, \Pi_M); s)$*, the following distributions are statistically indistinguishable:* $\mathcal{D}_0 = \{s' \xleftarrow{\$} \mathcal{R}_s : \mathsf{Sign}(\mathsf{sk}, (F, \Pi_M); s')\}$ *and* $\mathcal{D}_1 = \{s' \xleftarrow{\$} \mathcal{R}_s : \mathsf{Random}(\mathsf{vk}, (F, \Pi_M), \sigma; s')\}$*.*

The usual unforgeability notions apply (except strong unforgeability, since the signature is malleable, by definition). We now extend the randomization to signatures on randomizable ciphertexts:

Definition 6 (Randomizable Signature on Randomizable Ciphertexts).
Let $(\mathsf{Setup}, \mathsf{SKeyGen}, \mathsf{EKeyGen}, \mathsf{Encrypt}, \mathsf{Sign}, \mathsf{Decrypt}, \mathsf{Verif})$ *be a scheme of signatures on ciphertexts, with the following additional algorithm:*

- $\mathsf{Random}(\mathsf{vk}, \mathsf{pk}, c, \sigma; r', s')$ *outputs a ciphertext* c' *that encrypts the same message as* c *under the public key* pk*, and a signature* σ' *on* c'*. Further inputs are a signature* σ *on* c *under* vk*, and random coins* $r' \in \mathcal{R}_e$ *and* $s' \in \mathcal{R}_s$*.*

A signature on ciphertexts is called randomizable *if for any global parameters* $(\mathsf{param}_e, \mathsf{param}_s) \leftarrow \mathsf{Setup}(1^k)$*, keys* $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{EKeyGen}(\mathsf{param}_e)$ *and* $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param}_s)$*,* $m \in \mathcal{M}$*, and random coins* $r \in \mathcal{R}_e$ *and* $s \in \mathcal{R}_s$*, for* $c = \mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, m; r)$ *and* $\sigma = \mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c; s)$ *the following distributions* $\mathcal{D}_0$ *are statistically indistinguishable:*

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

We will denote by 1_e and 1_s the neutral elements in $\mathcal{R}_e$ and $\mathcal{R}_s$ that keep the ciphertexts and/or signatures unchanged after randomization. If $\mathcal{R}_e$ and $\mathcal{R}_s$ are groups (which will be the case for all our schemes, with addition being the group operation) and if we show that it is possible to additively update the randomness then this proves that the schemes are randomizable. The same unforgeability notion as above applies. If an additional extraction algorithm exists for the signature, we get extractable signatures on ciphertexts (defined below). Then, our above unforgeability notion for signatures on ciphertexts follows from the standard unforgeability notion on signatures.

2.4 Extractable Signatures on Randomizable Ciphertexts

For a scheme of signatures on randomizable ciphertexts (SRC) $\mathcal{SC}$, we define the following additional algorithm:

- $\mathsf{SigExt}_{\mathcal{SC}}(\mathsf{dk}, \mathsf{vk}, \sigma)$, which is given a decryption key, a verification key and a signature, outputs a signature σ'.

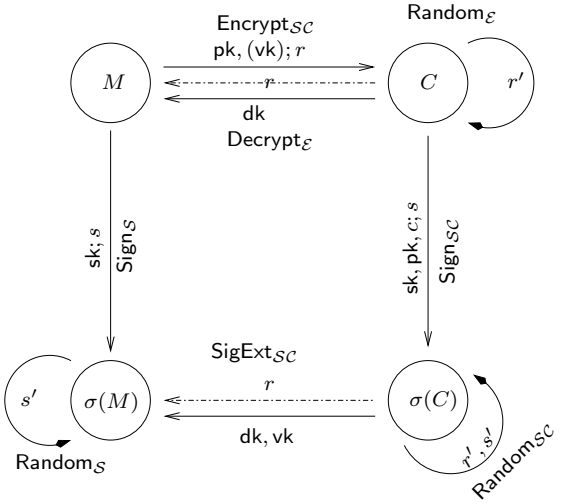
Let us assume that there is a signature scheme $\mathcal{S}$ where $\mathsf{Setup}_{\mathcal{S}}$ is the projection of $\mathsf{Setup}_{\mathcal{SC}}$ on the signature component and $\mathsf{SKeyGen}_{\mathcal{S}} = \mathsf{SKeyGen}$. The scheme $\mathcal{SC}$ is *extractable* if the following holds: for any $(\mathsf{param}_e, \mathsf{param}_s) \leftarrow \mathsf{Setup}_{\mathcal{SC}}(1^k)$, $(\mathsf{pk}, \mathsf{dk}) \leftarrow \mathsf{EKeyGen}(\mathsf{param}_e)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param}_s) = \mathsf{SKeyGen}_{\mathcal{S}}(\mathsf{param}_s)$, $m \in \mathcal{M}$, random coins $r \in \mathcal{R}_e, s \in \mathcal{R}_s$, for $c = \mathsf{Encrypt}_{\mathcal{SC}}(\mathsf{pk}, \mathsf{vk}, m; r)$ and $\sigma = \mathsf{Sign}_{\mathcal{SC}}(\mathsf{sk}, \mathsf{pk}, c; s)$, the output $\sigma' = \mathsf{SigExt}_{\mathcal{SC}}(\mathsf{dk}, \mathsf{vk}, \sigma)$ is a valid signature on m under vk, that is, $\mathsf{Verif}_{\mathcal{S}}(\mathsf{vk}, m, \sigma')$ is true.

An extractable SRC scheme $\mathcal{SC}$ allows the following: a user can encrypt a message m and obtain a signature σ on the ciphertext c. From (c, σ) the owner of the decryption key can now not only recover the encrypted message m, but also a signature σ' on the *message* m, using the functionality $\mathsf{SigExt}_{\mathcal{SC}}$. The signature σ on the ciphertext c could thus be seen as an *encryption of a signature* on the message m: for extractable signatures on ciphertexts, encryption and signing can thus be seen as commutative (see Figure 2).

2.5 Strong Extractability

We can immediately apply the notion of extractable signatures on randomizable ciphertexts to build a one-round classical blind signature scheme, but we can even consider more complex scenarios, such as *three-player blind signature schemes* (see Section 5) with applications to e-cash systems.

As already sketched above, we may have an additional property: as for encryption, knowing the random coins used for encryption may suffice to decrypt. After encrypting a message m as c, one knows the random



A message M can be encrypted using random coins r (Encrypt$_{\mathcal{SC}}$).
The signer can sign this ciphertext (Sign$_{\mathcal{SC}}$) and anyone can randomize the pair (Random$_{\mathcal{SC}}$).
A signature on the plaintext can be obtained using either dk (for SigExt$_{\mathcal{SC}}$) or the coins r (if $\sigma(C)$ has not been randomized); the result is the same as a signature of M by the signer (Sign$_{\mathcal{S}}$).

Fig. 2. (Strong) extractable signatures on randomizable ciphertexts

coins r used for the encryption. In all our instantiations we have that σ is the encryption of σ' with the same coins r used to encrypt the message. The user who encrypted m is thus able to extract σ', and not only the owner of the decryption key. A system $(\mathcal{SC}, \mathcal{S})$ with such a property will be called a *Strong Extractable (Randomizable) Signature on Ciphertexts* (augmented by the dotted lines in Figure 2).

3 A First Instantiation

Our first construction combines linear encryption [BBS04] and Waters signatures [Wat05] as follows: given an encryption of the "Waters hash" $\mathcal{F}(M)$ of a message M (and some additional values), the signer can make an encryption of a signature on M. Decrypting the latter leads thus to a classical Waters signature on M, which will provide extractability.

Before presenting the final scheme in Section 4, we first fix ideas by combining linear encryption and standard Waters signatures. We then modify the Waters signature to significantly improve efficiency of the scheme. The constructions we give here make all use of a symmetric pairing, whereas we give an instantiation for (more efficient) asymmetric pairings in Appendices C and D.

3.1 Assumptions

Our constructions rely on classical assumptions: CDH for the unforgeability of signatures and DLin for the semantic security of the encryption scheme, as well as soundness of the proofs:

Definition 7 (Computational Diffie-Hellman assumption (CDH)). *Let $\mathbb{G}$ be a cyclic group of prime order p. The* CDH *assumption in $\mathbb{G}$ states that for a generator g of $\mathbb{G}$ and random $a, b \in \mathbb{Z}_p$, given (g, g^a, g^b) it is hard to compute g^{ab}.*

Definition 8 (Decision Linear assumption (DLin)). *Let $\mathbb{G}$ be a cyclic group of prime order p. The* DLin *assumption states that given $(g, g^x, g^y, g^{xa}, g^{yb}, g^c)$ for random scalars $a, b, x, y, c \in \mathbb{Z}_p$, it is hard to decide whether $c = a + b$.*

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.

3.2 Basic Primitives

We briefly sketch the basic building blocks: commitments, linear encryption and the Waters signature. They are described in more detail in Appendix B. They need a pairing-friendly environment $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an admissible 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 basic descriptions, it follows immediately that the three primitives are randomizable.

Groth-Sahai Commitments. In the following, we will commit to values (group elements or scalars) and do proofs that they satisfy certain relations. We will use Groth-Sahai commitments that are secure under the DLin assumption: The commitment key is of the form $(\mathbf{u}_1 = (u_{1,1}, 1, g), \mathbf{u}_2 = (1, u_{2,2}, g), \mathbf{u}_3 = (u_{3,1}, u_{3,2}, u_{3,3})) \in (\mathbb{G}^3)^3$ and is set up by choosing $u_{1,1}, u_{2,2} \xleftarrow{\$} \mathbb{G}$ and $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p^*$ and setting $\mathbf{u}_3 = \mathbf{u}_1^\lambda \odot \mathbf{u}_2^\mu = (u_{3,1} = u_{1,1}^\lambda, u_{3,2} = u_{2,2}^\mu, u_{3,3} = g^{\lambda+\mu})$, which makes $\mathbf{u}_3$ a linear tuple w.r.t. $(u_{1,1}, u_{2,2}, g)$.

- To commit a group element $X \in \mathbb{G}$, choose random $s_1, s_2, s_3 \xleftarrow{\$} \mathbb{Z}_p$ and set

$$\mathcal{C}(X) := (1, 1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} \odot \mathbf{u}_3^{s_3} = (u_{1,1}^{s_1} \cdot u_{3,1}^{s_3}, u_{2,2}^{s_2} \cdot u_{3,2}^{s_3}, X \cdot g^{s_1+s_2} \cdot u_{3,3}^{s_3}).$$

- To commit a scalar $x \in \mathbb{Z}_p$, choose random coins $\gamma_1, \gamma_2 \in \mathbb{Z}_p$ and set

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

When $\mathbf{u}_3$ is a linear tuple these commitments are perfectly binding and the proofs will be perfectly sound. The committed values can even be extracted if the randomness of the commitment key is known (a scalar commitment allows extraction of x for small x only). However, if $\mathbf{u}_3$ is a random tuple (which is indistinguishable under DLin), the commitments become perfectly hiding and the proofs perfectly witness-indistinguishable.

Waters Signatures. The Waters signature scheme is formally described in Appendix B.2. The public parameters are a generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which defines the *Waters hash* of a message $M = (M_1, \ldots, M_k) \in \{0,1\}^k$ as $\mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$. A public key is of the form $\mathsf{vk} = Y = g^y$, with corresponding secret key $\mathsf{sk} = Z = h^y$, for a random $y \xleftarrow{\$} \mathbb{Z}_p$.

The signature on M is $\sigma = \big(\sigma_1 = Z \cdot \mathcal{F}(M)^s, \sigma_2 = g^{-s}\big)$, for some random $s \xleftarrow{\$} \mathbb{Z}_p$. It can be verified by checking $e(g, \sigma_1) \cdot e(\mathcal{F}(M), \sigma_2) = e(Y, h)$. We note that signing and verifying can be performed without knowing the message M itself; it suffices to know $F = \mathcal{F}(M)$. However, existential unforgeability [Wat05] (against chosen-message attacks under the CDH assumption) is for the pair (M, σ). As a consequence, if we work directly with $\mathcal{F}(M)$, we will need to add a proof of knowledge Π_M of M to guarantee unforgeability. Since our goal is to construct randomizable signatures and encryption, we will use Groth-Sahai proofs for a commitment C_M of M (bit-by-bit to make it extractable: $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k))$) and a proof that F is actually the evaluation of $\mathcal{F}$ on the committed M. Such a proof can be found in (the full version of) [FP09].

Linear Encryption. We formally describe linear encryption in Appendix B.3. The secret key dk is a pair of random scalars (x_1, x_2) and the public key is $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$. One encrypts a message $M \in \mathbb{G}$ as $c = \big(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot M\big)$, for random scalars $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$. To decrypt, one computes $M = c_3/(c_1^{1/x_1} c_2^{1/x_2})$. As shown by Boneh, Boyen and Shacham [BBS04], this scheme is semantically secure against chosen-plaintext attacks (IND-CPA) under the DLin assumption.

3.3 Waters Signature on Linear Ciphertexts

Using Waters signatures, we will sign a linear encryption of $F = \mathcal{F}(M)$. We note that from a "ciphertext" using the decryption key, one can only extract $\mathcal{F}(M)$ (from which M can be obtained for small message spaces). As mentioned before, signatures remain unforgeable on F if in addition a proof Π_M of knowledge of M such that $F = \mathcal{F}(M)$ is given. The keys are independent Waters signature keys ($\mathsf{vk} = Y = g^y$ and $\mathsf{sk} = Z = h^y$), and linear encryption keys ($\mathsf{dk} = (x_1, x_2)$, $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$). A first idea would be to define a signature on an encrypted message $c = \big(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \mathcal{F}(M)\big)$ as $\sigma = (c_1^s, c_2^s, Z \cdot c_3^s)$. However, there are two problems:

- While the randomization of the signing coins s into $s + s'$ is easy from c, the randomization of the encryption coins r into $r + r'$ requires the knowledge of the values X_1^s, X_2^s and g^s (see Section 4.2 for how to randomize). We therefore include them in the signature.
- For the reduction of our notion of unforgeability to the security of Waters' scheme, we need to simulate the oracle returning signatures on ciphertexts having a Waters signature oracle. We can first extract M from the proof of knowledge Π_M and submit M to our oracle. From a reply $(Z \cdot \mathcal{F}(M)^s, g^{-s})$, we then have to generate $\sigma = (c_1^s, c_2^s, Z \cdot c_3^s; X_1^s, X_2^s, g^s)$ for an unknown s. We could do so if we knew the randomness (r_1, r_2) for c_1, c_2 and c_3; hence we add another proof to the extended ciphertext: Π_r proves knowledge of r_1 and r_2, used to encrypt $\mathcal{F}(M)$, which consists of bit-by-bit commitments $C_1 = (\mathcal{C}'(r_{1,1}), \ldots, \mathcal{C}'(r_{1,\ell}))$ and $C_2 = (\mathcal{C}'(r_{2,1}), \ldots, \mathcal{C}'(r_{2,\ell}))$, where ℓ is the bit-length of the order p, and proofs that each sub-commitment is indeed a bit commitment.

The global proof on the message and the randomness, which we denote by $\Pi = (\Pi_M, \Pi_r)$, can be done with randomizable commitments and proofs, using the Groth-Sahai methodology [GS08, FP09], and consists of $9k + 18\ell + 6$ group elements (where k and ℓ are the respective bit lengths of messages and of the order of $\mathbb{G}$). Such an extended ciphertext (c, Π) can then be signed, after a test of validity of the proof Π. Decryption and verification follow straight from the corresponding algorithms for Waters signatures and linear encryption. More interestingly, the above signature on randomizable ciphertexts is *extractable*: on a valid signature, if one knows the decryption key $\mathsf{dk} = (x_1, x_2)$, one can compute $\Sigma = (\Sigma_1 = \sigma_3/(\sigma_1^{1/x_1}\sigma_2^{1/x_2}), \Sigma_2 = \sigma_6^{-1})$, which is a valid signature on M:

$$\Sigma_1 = \sigma_3/(\sigma_1^{1/x_1}\sigma_2^{1/x_2}) = Z \cdot g^{s(r_1+r_2)} \cdot \mathcal{F}(M)^s/(g^{sr_1}g^{sr_2}) = Z \cdot \mathcal{F}(M)^s$$
$$\Sigma_2 = \sigma_6^{-1} = g^{-s}$$

Note that without knowing the decryption key, the same can be obtained from the coins (r_1, r_2) used for encryption: $\Sigma = (\Sigma_1 = \sigma_3/\sigma_6^{r_1+r_2}, \Sigma_2 = \sigma_6^{-1})$.

From the randomization formula of the basic schemes, we easily get the randomization property of the above Waters signature on linear ciphertexts. One shows unforgeability in the UF sense under the CDH assumption in $\mathbb{G}$: extractability provides a forgery on a new message, but only known as $F = \mathcal{F}(M)$. Since one also has to provide a valid proof Π_M that contains commitments to the bits of message M, the knowledge of the trapdoor (λ, μ) for the commitments allows to recover M too, which leads to an existential attack of the basic Waters signature scheme. The complete description and security analysis can be found in Appendix B.

4 An Efficient Instantiation

The construction in the previous section is a concrete and feasible signature on randomizable ciphertexts, which is furthermore extractable, and even in a strong way. We have thus achieved our goal, and all the applications we had in mind can benefit from it. The main drawback, from an efficiency point of view, are the bit-by-bit commitments C_M, C_1 and C_2 of M, r_1 and r_2, respectively. Whereas the message M to be signed

could be short (and even a single bit for voting schemes), r_1 and r_2 are necessarily large (the bit length of the order of the group). For a k-bit long message M, Π_M (composed of C_M and a proof) consists of $9k+2$ group elements. The random coins r_1 and r_2 being ℓ-bit long, Π_r (which includes C_1, C_2 and the proof) requires $18\ell+4$ group elements. We now revisit the Waters signature scheme, which will allow us to remove the costly bit-by-bit commitments C_1 and C_2.

The main idea for the construction is to build a scheme which is unforgeable against a stronger kind of chosen-message attack under the same assumption: the adversary can submit "extended messages" $(M, R_1 := g^{r_1}, R_2 := g^{r_2}, Y_1 = Y^{r_1}, Y_2 = Y^{r_2})$ and the oracle replies with the tuple $(\mathsf{sk} \cdot (\mathcal{F}(M)R_1R_2)^s, g^{-s}, R_1^{-s}, R_2^{-s})$. We name this attack *chosen-extended-message attack* and note that this security notion implies the classical one, since querying $(M, 1_{\mathbb{G}}, 1_{\mathbb{G}}, \mathsf{vk})$ yields a signature on M. Intuitively, the extra parameters (R_1, R_2) will allow simulation of the signature on the ciphertext without having to know the random coins r_1 and r_2 explicitly.

4.1 Revisited Waters Signature

Our variant is defined by the four algorithms.

- $\mathsf{Setup}(1^k)$: The scheme is defined over a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, where $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is an admissible bilinear map, $\mathbb{G}$ and $\mathbb{G}_T$ are groups of prime order p, generated by g and $e(g,g)$ respectively. We will sign messages $M = (M_1, \ldots, M_k) \in \{0,1\}^k$. The parameters are a randomly chosen generator $h \xleftarrow{\$} \mathbb{G}$ and a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, which defines the *Waters Hash* as $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. We set $\mathsf{param} := (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$.
- $\mathsf{SKeyGen}(\mathsf{param})$: Choose a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key $\mathsf{vk} = Y = g^y$, and the secret key as $\mathsf{sk} = Z = h^y$.
- $\mathsf{Sign}(\mathsf{sk} = Z, M, R_1, R_2, Y_1, Y_2; s)$: First check the consistency of (R_1, R_2, Y_1, Y_2): if $e(R_1, Y) = e(g, Y_1), e(R_2, Y) = e(g, Y_2)$ then this guarantees that there exists (r_1, r_2) such that $R_i = g^{r_i}, Y_i = Y^{r_i}$. Choose a random $s \xleftarrow{\$} \mathbb{Z}_p$ and define the signature as $\sigma = \big(\sigma_1 = Z \cdot (\mathcal{F}(M)R_1R_2)^s, \sigma_2 = g^{-s}, \sigma_3 = R_1^{-s}, \sigma_4 = R_2^{-s}\big)$. Again, we may replace the input message M by the pair $(\mathcal{F}(M), \Pi_M)$.
- $\mathsf{Verif}(\mathsf{vk} = Y, M, R_1, R_2, Y_1, Y_2, \sigma)$: Check whether $e(g, \sigma_1) \cdot e(\mathcal{F}(M)R_1R_2, \sigma_2) = e(Y, h), e(g, \sigma_3) = e(\sigma_2, R_1)$ and $e(g, \sigma_4) = e(\sigma_2, R_2)$, as well as the consistency of (R_1, R_2, Y_1, Y_2).

To randomize a signature, we define $\mathsf{Random}(\mathsf{vk}, (F, \Pi_M), R_1, R_2, Y_1, Y_2, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4); s')$ to output $\sigma' = (\sigma_1 \cdot (FR_1R_2)^{s'}, \sigma_2 \cdot g^{-s'}, \sigma_3 \cdot R_1^{-s'}, \sigma_4 \cdot R_2^{-s'})$, for a random $s' \xleftarrow{\$} \mathbb{Z}_p$. This simply changes the initial randomness s to $s + s' \bmod p$. Hence, if s' is uniform then the internal randomness of σ' is uniform in $\mathbb{Z}_p$.

Theorem 9. *Our variant of the Waters signature scheme is randomizable, and existentially unforgeable under chosen-extended-message attacks if the* CDH *assumption holds.*

The proof of unforgeability is similar to that for the original Waters scheme and can be found in Appendix A.

4.2 Signatures on Encrypted Messages

In our new scheme, we will sign a linear encryption of $F = \mathcal{F}(M)$ using our Revisited Waters signatures:

- $\mathsf{Setup}(1^k)$: The scheme is based on a bilinear group $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, which constitutes the parameters param_e for encryption. For the signing part, we require moreover a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, and a generator $h \xleftarrow{\$} \mathbb{G}$ and define $\mathsf{param}_s := (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$.
- $\mathsf{EKeyGen}(\mathsf{param}_e)$: Choose two random scalars $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, which define the secret key $\mathsf{dk} = (x_1, x_2)$, and the public key as $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$.

– $\mathsf{SKeyGen}(\mathsf{param}_s)$: Choose a random scalar $y \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = Y = g^y$, and the secret key as $\mathsf{sk} = Z = h^y$.
– $\mathsf{Encrypt}\big(\mathsf{pk} = (X_1, X_2), \mathsf{vk} = Y, M; (r_1, r_2)\big)$: For a message $M \in \{0,1\}^k$ and random scalars $r_1, r_2 \in \mathbb{Z}_p$, define the ciphertext as $c = \big(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \mathcal{F}(M)\big)$. To guarantee our notion of unforgeability of signatures on ciphertexts, we add proofs of knowledge of M and an image of r_1 and r_2:
  - Proof Π_r contains the commitments $C_r = (C_1 = \mathcal{C}(Y_1^r), C_2 = \mathcal{C}(Y_2^r))$, from which the simulator can extract Y_1, Y_2 in the reduction (see below). Π_r moreover contains proofs of consistency: $e(\langle C_i \rangle, X_i) = e(c_i, Y)$. These equations are linear pairing product equations. We require 6 group elements for the commitments and 6 for the proofs, thus 12 group elements instead of $18\ell + 4$ in the previous construction.
  - Proof Π_M proves knowledge of M s.t. $\mathcal{F}(M)$ is encrypted in c. It 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 ($6k$ group elements); moreover, a proof that c_3 is well-formed: $e(c_3, Y) = e(u_0 \prod_{i \in \{1,\ldots,k\}} u_i^{\langle M_i \rangle}), Y) \cdot e(\langle C_1 \rangle \langle C_2 \rangle, g)$, which is a linear pairing product equation (3 additional group elements). Π_M is therefore composed of $9k + 3$ group elements.

  The global proof (containing the commitments) Π consists therefore of $9k + 15$ group elements (instead of $9k + 18\ell + 6$ when using the original Waters scheme), where k and ℓ are the bit lengths of the message M and elements of $\mathbb{G}$, respectively.
– $\mathsf{Sign}\big(\mathsf{sk} = Z, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi); s\big)$: To sign a ciphertext $c = (c_1, c_2, c_3)$, first check if Π is valid, and if so, output

$$\sigma = (c_1^s,\ c_2^s,\ Z \cdot c_3^s;\ X_1^s,\ X_2^s, g^s)\ .$$

– $\mathsf{Decrypt}\big(\mathsf{dk} = (x_1, x_2), \mathsf{vk} = Y, (c = (c_1, c_2, c_3), \Pi)\big)$: On a valid ciphertext (verifiable via Π), knowing the decryption key $\mathsf{dk} = (x_1, x_2)$, one can obtain $F = \mathcal{F}(M)$ since $F = c_3/(c_1^{1/x_1} c_2^{1/x_2})$.
– $\mathsf{Verif}\big(\mathsf{vk} = Y, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi), \sigma = (\sigma_1, \sigma_2, \sigma_3; \sigma_4, \sigma_5, \sigma_6)\big)$: In order to verify the signature, one verifies Π and checks whether the following pairing equations hold: $e(\sigma_3, g) = e(h, Y) \cdot e(c_3, \sigma_6)$ and

$$\begin{aligned} e(\sigma_1, X_1) &= e(c_1, \sigma_4) & \qquad e(\sigma_2, X_2) &= e(c_2, \sigma_5) \\ e(\sigma_1, g) &= e(c_1, \sigma_6) & \qquad e(\sigma_2, g) &= e(c_2, \sigma_6) \end{aligned}$$

– $\mathsf{Random}\big(\mathsf{vk} = Y, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi), \sigma; r_1', r_2', s'\big)$: In order to randomize the signature and the ciphertext, the algorithm outputs:

$$\begin{aligned} c' &= \big(c_1 \cdot X_1^{r_1'}, c_2 \cdot X_2^{r_2'}, c_3 \cdot g^{r_1'+r_2'}\big) \\ \sigma' &= \big(\sigma_1 \cdot c_1^{s'} \cdot \sigma_4^{r_1'} \cdot X_1^{r_1' s'}, \sigma_2 \cdot c_2^{s'} \cdot \sigma_5^{r_2'} \cdot X_2^{r_2' s'}, \sigma_3 \cdot c_3^{s'} \cdot \sigma_6^{r_1'+r_2'} \cdot g^{(r_1'+r_2')s'} \\ &\qquad \sigma_4 \cdot X_1^{s'}, \sigma_5 \cdot X_2^{s'}, \sigma_6 \cdot g^{s'}\big) \end{aligned}$$

together with a randomization Π' of Π.
– $\mathsf{SigExt}\big(\mathsf{dk} = (x_1, x_2), \mathsf{vk}, (c = (c_1, c_2, c_3), \Pi), \sigma\big)$: Return the following: $\Sigma = \big(\Sigma_1 = \sigma_3/(\sigma_1^{1/x_1} \sigma_2^{1/x_2}), \Sigma_2 = \sigma_6^{-1}\big)$, which is a valid signature on M:

$$\begin{aligned} \Sigma_1 &= \sigma_3/(\sigma_1^{1/x_1} \sigma_2^{1/x_2}) = Z \cdot g^{s(r_1+r_2)} \cdot \mathcal{F}(M)^s / g^{s r_1} g^{s r_2} = Z \cdot \mathcal{F}(M)^s, \\ \Sigma_2 &= \sigma_6^{-1} = g^{-s}. \end{aligned}$$

The same can be obtained from the coins (r_1, r_2) used for encryption.

Theorem 10. *The above scheme is randomizable and unforgeable (in the UF sense) under the* CDH *assumption in* $\mathbb{G}$.

Proof. Correctness of Random follows from inspection of the construction of c' and σ' and the fact that Groth-Sahai proofs are randomizable.

Since we have proved that our variant of Waters signatures is secure under a stronger kind of attack, we can use it for an appropriate simulation of the signing oracle. The full proof can be found in Appendix A. As motivated when introducing the additional elements in the signature query, from a valid signing query, our simulator can extract M, but also $R_i = g^{r_i}$, $Y_i = Y^{r_i}$ (but not the scalars r_1 and r_2), partly from the commitment and partly using dk. It adds M to the set SM and then queries $\mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}, M, R_1, R_2, Y_1, Y_2)$ to the extended-message signing oracle to obtain $\sigma' = (\sigma'_1 = \mathsf{sk} \cdot (\mathcal{F}(M)R_1R_2)^s, \sigma'_2 = g^{-s}, \sigma'_3 = R_1^{-s}, \sigma'_4 = R_2^{-s})$. It then returns the following to the adversary:

$$\sigma = \begin{pmatrix} \sigma_1 = {\sigma'_3}^{-x_1} = g^{sr_1x_1} = X_1^{sr_1}, \sigma_2 = {\sigma'_4}^{-x_2} = g^{sr_2x_2} = X_2^{sr_2}, \sigma_3 = \sigma'_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s \cdot g^{s(r_1+r_2)} \\ \sigma_4 = {\sigma'_3}^{-x_1} = X_1^s, \qquad \sigma_5 = {\sigma'_4}^{-x_2} = X_2^s, \qquad \sigma_6 = {\sigma'_2}^{-1} = g^s \end{pmatrix} .$$

Finally, if the adversary wins by outputting a ciphertext and a signature, we can extract a signature on the plaintext and the plaintext from the proof of knowledge. □

5 Applications

We have introduced *extractable signatures on randomizable ciphertexts* (ESRC), a new primitive that has many applications to anonymity. A first straightforward application is to blind signatures, which yields a similar (yet more efficient) result to [MSF10, GK08]; however, this does not exploit all the power of our new tool. A more interesting application is to receipt-free voting schemes. We discuss this in the following and then show how to construct variants of blind signatures from our primitive.

5.1 Non-interactive Receipt-Free E-voting

In voting schemes, anonymity is a crucial property: nobody should be able to learn the content of my vote. This can be achieved with encryption schemes. However, this does not address the problem of *vote sellers*: a voter may sell his vote and then reveal/prove the content of his encrypted vote to the buyer. He could do so by simply revealing the randomness used when encrypting the vote, which allows to verify that a claimed message was encrypted.

A classical approach to prevent vote selling uses heavy interactive techniques based on randomizable encryption schemes and designated-verifier zero-knowledge proofs: the voter encrypts his vote v as c and additionally signs it to bar any modification by the voting center. But before doing so, the voting center randomizes c into c' (which cannot be opened by the voter anymore since he no longer knows the random coins) and then proves that c and c' contain the same plaintext. This proof must be non-transferable, otherwise the voter could open c (by revealing the random coins) and transfer the proof to the buyer, which together yields a proof of opening for c'. The used proof is thus a designated-verifier zero-knowledge proof. Finally, after receiving c' and being convinced by the proof, the voter signs c'.

Signatures on ciphertexts that can be randomized allow to avoid interactions altogether: all a voter does is encrypt his vote v as c and make a signature σ on c. The voting center can now consistently randomize both c and σ as c' and σ', so that the randomness used in c' is unknown to the signer, who is however guaranteed that the vote was not modified by the voting center because of the unforgeability notion for ESRC: nobody can generate a signature on a ciphertext that contains a different plaintext. We have thus constructed a non-interactive *receipt-free* voting scheme.

Since our ESRC candidates use not only randomizable but homomorphic encryption schemes (the encryption of the vote is actually the bit-commitments of the M_i's, which are either linear encryptions or ElGamal encryptions of g^{M_i}), classical techniques for voting schemes with homomorphic encryption and threshold

decryption can be used [BFP+01]: there is no risk for the signature on the ciphertext to be converted into a signature on the plaintext if the board of authorities uses the decryption capability on the encrypted tally only.

If the vote consists of one box to be checked, the size of the ballot is only 33 group elements in the instantiation with linear encryption, and even smaller for the instantiation using ElGamal detailed in Appendix D: 15 $\mathbb{G}_1$ elements and 9 $\mathbb{G}_2$ elements. Furthermore, if the vote consists of several (say k) boxes to be checked or not, with various constraints, the ballot size grows only slowly in k, since while the votes are committed bit by bit, the proofs can be global. Hence, the size basically corresponds to the signature on a ciphertext of a k-bit message. The extended ciphertexts already contains proofs that plaintexts are bits only, and all the proofs are randomizable.

5.2 Blind Signatures and Variants

Since the beginning of e-cash, blind signatures have been their most important tool. They provide an interactive protocol between a bank and a user, letting a user have a message signed by the bank without revealing it. Moreover, the message-signature pair obtained by the user is uncorrelated to the view of the protocol execution by the bank, which enables the user to withdraw anonymous coins. Several signature schemes have been turned into blind signature schemes. The best-known is the first scheme by Chaum [Cha83], which is derived from RSA signatures [RSA78], and has been proven secure [BNPS01] under the one-more RSA assumption in the random-oracle model [BR93]. As defined in [PS96,PS00], for e-cash, the security requirement is the resistance to one-more forgeries: after interacting q times with the signer, an adversary should not be able to output more than q valid signed messages.

With ESRC, one can build a computationally blind signature scheme: the user encrypts the message m into the ciphertext c under his own key, and asks for a signature on c. He gets back a signature on the ciphertext c from which he can then extract σ, a valid signature on m. This signature is not yet blind, since the signer knows the coins used to compute it, and can thus link σ to the transcript. However, due to the randomizability of the signature, the user can randomize σ into σ' that is a secure blind signature:

- the blindness property relies on the semantic security of the encryption scheme (here DLin) and the randomization, which is information-theoretic;
- the one-more unforgeability relies on unforgeability of the signature scheme (here CDH), since the user cannot generate a signature for a message that has not been asked, encrypted, to the signer. Of course, we do not obtain strong one-more unforgeability (where several signatures on the same message would be counted several times), which is impossible with randomizable signatures.

This construction is similar to [GK08] but with better efficiency and much less bandwidth consumption since the latter relies on inefficient NIZK techniques [DFN06].

One-Round Fair Blind Signatures. With a *strong* extractable randomizable signature on ciphertexts, we get more than just standard blind signatures: we have *fair* blind signatures [SPC95]. Using a *strong* ESRC scheme, the user does not need to encrypt m under his own key, since the random coins suffice to extract the signature. He can thus encrypt the message m under a tracing authority's key. Using the decryption key, the authority can extract the message from c (or at least check if c encrypts a purported message) w.r.t. the signed message and thus revoke anonymity in case of abuse.

One-Round Three-Party Blind Signatures. Our primitive also allows to design a *three-party* blind signature scheme, which we define as follows: a party A makes a signer C sign a message m for B so that neither A nor C can later link the final message-signature pair (for A among all the signatures for the message m, and for C among all the valid message-signature pairs). To realize this primitive, the party A encrypts

the message m under the key of B, and sends it to the signer C, who signs the ciphertext and applies the randomization algorithm to the ciphertext-signature pair (this is useful only in case A and B are distinct, as then A does not know the randomness for encryption and therefore cannot extract a signature). C sends the encrypted signature to B (possibly via A, who cannot decrypt anyway) and B also applies the randomization algorithm (so that C does not know the random coins used for signing) and then extracts the signature. With such a 2-flow scheme, B can obtain a signature, unknown to A, on a message chosen by A, unknown and even indistinguishable from any message-signature pair to C. Applied to group signatures, such a primitive allows a group manager A to add a new member B without learning his certificate provided by the authority C: A can define the rights in the message, but only B receives the certificate generated by C.

Additional Properties. Using our instantiation of ESRC, we can define an additional trapdoor: the extraction key for the commitments. It is not intended to be known by anybody (except the simulator in the security analysis), since the commitment key is in the CRS, but one could consider a scenario where it is given to a trusted authority that gets revocation capabilities.

Our construction is similar to previous efficient round-optimal blind signatures [Fuc09, AFG+10] in that it uses Groth-Sahai proofs. However, we rely on standard assumptions only, and our resulting blind signature is a standard Waters signature, which is much shorter (2 group elements!) than the proof of knowledge of a signature used in all previous constructions.

Acknowledgments

This work was supported by the French ANR-07-TCOM-013-04 PACE Project, by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II and by EADS.

References

[AFG+10] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In T. Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 209–236. Springer, Aug. 2010.

[BBS04] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In M. Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, Aug. 2004.

[BCC+09] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In S. Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 108–125. Springer, Aug. 2009.

[BFP+01] O. Baudron, P.-A. Fouque, D. Pointcheval, J. Stern, and G. Poupard. Practical multi-candidate election system. In *20th ACM Symposium Annual on Principles of Distributed Computing*, pages 274–283. ACM Press, Aug. 2001.

[BNPS01] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The power of RSA inversion oracles and the security of Chaum's RSA-based blind signature scheme. In P. F. Syverson, editor, *FC 2001: 5th International Conference on Financial Cryptography*, volume 2339 of *Lecture Notes in Computer Science*, pages 319–338. Springer, Feb. 2001.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73. ACM Press, Nov. 1993.

[BW06] X. Boyen and B. Waters. Compact group signatures without random oracles. In S. Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, May / June 2006.

[Cha83] D. Chaum. Blind signatures for untraceable payments. In D. Chaum, R. L. Rivest, and A. T. Sherman, editors, *Advances in Cryptology – CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1983.

[DFN06] I. Damgård, N. Fazio, and A. Nicolosi. Non-interactive zero-knowledge from homomorphic encryption. In S. Halevi and T. Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 41–59. Springer, Mar. 2006.

[Fis06] M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In C. Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77. Springer, Aug. 2006.

[FP09] G. Fuchsbauer and D. Pointcheval. Proofs on encrypted values in bilinear groups and an application to anonymity of signatures. In H. Shacham and B. Waters, editors, *PAIRING 2009: 3rd International Conference on Pairing-based Cryptography*, volume 5671 of *Lecture Notes in Computer Science*, pages 132–149. Springer, Aug. 2009.

[Fuc09] G. Fuchsbauer. Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. Cryptology ePrint Archive, Report 2009/320, 2009. `http://eprint.iacr.org/`.

[Fuc10] G. Fuchsbauer. Commuting signatures and verifiable encryption and an application to non-interactively delegatable credentials. Cryptology ePrint Archive, Report 2010/233, 2010. `http://eprint.iacr.org/`.

[GK08] K. Gjøsteen and L. Kråkmo. Round-optimal blind signatures from Waters signatures. In J. Baek, F. Bao, K. Chen, and X. Lai, editors, *ProvSec*, volume 5324 of *Lecture Notes in Computer Science*, pages 112–126. Springer, 2008.

[GMR88] S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, Apr. 1988.

[GS08] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In N. P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, Apr. 2008.

[HK08] D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In D. Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, Aug. 2008.

[MSF10] S. Meiklejohn, H. Shacham, and D. M. Freeman. Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures. In M. Abe, editor, *Proceedings of Asiacrypt 2010*, volume 6477 of *Lecture Notes in Computer Science*, pages 519–538. Springer, 2010.

[PS96] D. Pointcheval and J. Stern. Provably secure blind signature schemes. In K. Kim and T. Matsumoto, editors, *Advances in Cryptology – ASIACRYPT'96*, volume 1163 of *Lecture Notes in Computer Science*, pages 252–265. Springer, Nov. 1996.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, 2000.

[RSA78] R. L. Rivest, A. Shamir, and L. M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

[SPC95] M. Stadler, J.-M. Piveteau, and J. Camenisch. Fair blind signatures. In L. C. Guillou and J.-J. Quisquater, editors, *Advances in Cryptology – EUROCRYPT'95*, volume 921 of *Lecture Notes in Computer Science*, pages 209–219. Springer, May 1995.

[Wat05] B. R. Waters. Efficient identity-based encryption without random oracles. In R. Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127. Springer, May 2005.

A Security Proofs for the Revisited Waters Signature on Linear Ciphertexts

Theorem 9. *Our variant of the Waters signature scheme is existentially unforgeable under chosen-extended-message attacks if the* CDH *assumption holds.*

Proof. Let $\mathcal{A}$ be an adversary breaking the existential unforgeability of the above signature scheme, i.e. after at most q_s signing queries, it succeeds in building a new signature with probability ϵ. Let $(g, A = g^a, B = g^b)$ be a CDH-instance. We show how an adversary $\mathcal{B}$ can compute g^{ab} thanks to $\mathcal{A}$.

$\mathsf{Setup}_{\mathcal{S}}$. Pick a random position $j \xleftarrow{\$} \{0, \ldots, k\}$, choose random indices $y_0, y_1, \ldots, y_k \xleftarrow{\$} \{0, \ldots, 2q_s - 1\}$, and random scalars $z_0, z_1, \ldots, z_k \xleftarrow{\$} \mathbb{Z}_p$. One defines $Y = A = g^a$, $h = B = g^b$, $u_0 = h^{y_0 - 2jq_s} g^{z_0}$, $u_i = h^{y_i} g^{z_i}$.

Signing queries. To answer a signing query on a message $M = (M_i)$, we define

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

If $H \equiv 0 \pmod p$ then abort, otherwise set $\sigma = (Y^{-J/H}(Y_1 Y_2)^{-1/H}(\mathcal{F}(M) R_1 R_2)^s, Y^{1/H} g^{-s}, Y_1^{1/H} R_1^{-s}, Y_2^{1/H} R_2^{-s})$. Defining $\tilde{s} = s - a/H$, we have:

$$\begin{aligned}
\sigma &:= \big(Y^{-J/H}(Y_1 Y_2)^{-1/H}(h^H g^J R_1 R_2)^s,\, Y^{1/H} g^{-s},\, Y_1^{1/H} R_1^{-s},\, Y_2^{1/H} R_2^{-s}\big) \\
&= \big(Y^{-J/H} Y^{-(r_1+r_2)/H}(h^a g^{Ja/H} g^{(r_1+r_2)a/H})(\mathcal{F}(M) R_1 R_2)^{\tilde{s}},\, Y^{1/H} g^{-a/H} g^{-\tilde{s}},\, Y_1^{1/H} R_1^{-s},\, Y_2^{1/H} R_2^{-s}\big) \\
&= \big(h^a (\mathcal{F}(M) R_1 R_2)^{\tilde{s}},\, g^{-\tilde{s}},\, R_1^{-\tilde{s}},\, R_2^{-\tilde{s}}\big).
\end{aligned}$$

After at most q_s signing queries $\mathcal{A}$ outputs a forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)$ on M^*. As before, we define

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

If $H^* \not\equiv 0 \pmod p$ then abort, otherwise, for some s^*, $\sigma^* = (h^a(\mathcal{F}(M^*)R_1R_2)^{s^*}, g^{-s^*}, R_1^{-s^*}, R_2^{-s^*})$, and thus $\sigma^* = (h^a g^{J^* s^*}(R_1R_2)^{s^*}, g^{-s^*}, R_1^{-s^*}, R_2^{-s^*})$. As a consequence, $\sigma_1^*(\sigma_2^*)^{J^*}\sigma_3\sigma_4 = h^a = g^{ab}$: one has solved the CDH problem.

Success Probability. (Based on [HK08]) The Waters hash function is $(1, \delta)$-programmable (i.e. we can find with non negligible probability a case where δ intermediate hashes are not null, and the last one is), therefore the previous simulation succeeds with non negligible probability ($\Theta(\epsilon/q_s\sqrt{k})$), and so $\mathcal{B}$ breaks CDH. □

Theorem 10. *Our variant of Waters signature on linear ciphertexts is unforgeable (in the UF sense) under the* CDH *assumption in* $\mathbb{G}$.

Proof. Let us denote $\mathcal{SC}$ our above signature on ciphertexts (but omit it in the subscripts for clarity), and $\mathcal{S}$ our variant of Waters signature scheme. We know that the latter is existentially unforgeable against chosen-extended-message attacks under the CDH assumption. Let us assume that $\mathcal{A}$ is able to break the unforgeability of $\mathcal{SC}$. We will build an adversary $\mathcal{B}$ against our variant of Waters signature scheme. We note that $\mathcal{B}$ generated the parameters for the commitments for the proof Π of knowledge of M, g^{r_1} and g^{r_2}, so that it can extract the values.

- $\mathsf{Setup}(1^k)$: we first run the $\mathsf{Setup}_{\mathcal{S}}(1^k)$ algorithm, from which we get $\mathsf{param}_{\mathcal{S}} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$. We set $\mathsf{param}_s = \mathsf{param}_{\mathcal{S}} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$, and $\mathsf{param}_e = (p, \mathbb{G}, \mathbb{G}_T, e, g)$. $\mathcal{B}$ sets the commitment parameters so that it can extract committed values.
- $\mathsf{EKeyGen}(\mathsf{param}_e)$: for each new key request, $\mathcal{B}$ chooses two random scalars $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, which define the secret key $\mathsf{dk} = (x_1, x_2)$, and the public key as $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$.
- $\mathsf{SKeyGen}(\mathsf{param}_s)$: for the unique signing key request, one gets the verification key $\mathsf{vk}_{\mathcal{S}}$ from our variant of the Waters unforgeability security game. $\mathcal{B}$ sets $\mathsf{vk} = X = \mathsf{vk}_{\mathcal{S}}$.
- $\mathcal{A}$ can now access a signing oracle, with queries of the form $\mathsf{Sign}(\mathsf{vk}, \mathsf{pk}, \cdot)$, for any pk and ciphertext of its choice. But the ciphertext looks like $c = (c_1, c_2, c_3)$ together with Π, that contains C_M (with extractable M), $C_r = (C_1, C_2, C_3)$ (with extractable g^{r_1}, g^{r_2} and $X^{r_1+r_2}$).
  - If the tuple (c, Π) is not valid, then $\mathcal{B}$ returns $\perp$;
  - Otherwise, $\mathcal{B}$ can extract M from the bit-by-bit proof of knowledge Π_M, as well as $Y_1 = X^{r_1}$, $Y_2 = X^{r_2}$ from C_r and using dk, $R_1 = c_1^{x_1}, R_2 = c_2^{r_2}$. It then queries $\mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}_{\mathcal{S}}, M, R_1, R_2, Y_1, Y_2)$ to the extended-message signing oracle, and adds (vk, M) to the SM set. It receives back $\sigma' = (\sigma'_1 = \mathsf{sk} \cdot (\mathcal{F}(M)R_1R_2)^s, \sigma'_2 = g^{-s}, \sigma'_3 = R_1^{-s}, \sigma'_4 = R_2^{-s})$.
    It then returns

$$\sigma = \begin{pmatrix} \sigma_1 = {\sigma'_3}^{-x_1} = g^{sr_1x_1} = Y_1^{sr_1}, \sigma_2 = {\sigma'_4}^{-x_2} = g^{sr_2x_2} = Y_2^{sr_2}, \sigma_3 = \sigma'_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s \cdot g^{s(r_1+r_2)}, \\ \sigma_4 = {\sigma'_2}^{-x_1} = g^{sx_1} = X_1^s, \quad \sigma_5 = {\sigma'_2}^{-x_2} = g^{sx_2} = X_2^s, \quad \sigma_6 = {\sigma'_2}^{-1} = g^s \end{pmatrix}.$$

- After a polynomial number of queries, $\mathcal{A}$ outputs, with non-negligible probability, a valid signature σ on a valid ciphertext (c, Π). As above, from Π and the extraction key, $\mathcal{B}$ can extract the message M. For a valid forgery, one needs $M \neq \perp$ and $(M, \mathsf{vk}) \notin \mathsf{SM}$. Again, from Π, the extraction key and the commitment key, $\mathcal{B}$ can obtain (R_1, R_2, Y_1, Y_2). One sets

$$\sigma_{\mathcal{B}} = (\Sigma_1 = \mathsf{sk}(\mathcal{F}(M)R_1R_2)^s, \quad \Sigma_2 = \sigma_6^{-1} = g^{-s}, \quad \Sigma_3 = \sigma_1^{1/x_1} = R_1^s, \quad \Sigma_4 = \sigma_2^{1/x_2} = R_2^s).$$

  And so $(M, R_1, R_2, Y_1, Y_2, \sigma_{\mathcal{B}})$ is a valid forgery. This breaks the security of our variant of the Waters signature scheme, that holds under the CDH assumption.

□

B Basic Waters Signature on Linear Ciphertexts

B.1 Groth-Sahai Commitments

In the following, several elements (group elements or scalars) will have to be committed so that proofs can be done on them. We will use Groth-Sahai commitments that are secure under the DLin assumption: they 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 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. The commitment key is of the form $(\mathbf{u}_1 = (u_{1,1}, 1, g), \mathbf{u}_2 = (1, u_{2,2}, g), \mathbf{u}_3 = (u_{3,1}, u_{3,2}, u_{3,3})) \in (\mathbb{G}^3)^3$. Initialization of the parameters should be: $\mathbf{u}_3 = \mathbf{u}_1^\lambda \odot \mathbf{u}_2^\mu = (u_{3,1} = u_{1,1}^\lambda, u_{3,2} = u_{2,2}^\mu, u_{3,3} = g^{\lambda+\mu})$ with $\lambda, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, and random elements $u_{1,1}, u_{2,2} \xleftarrow{\$} \mathbb{G}$, which means that $\mathbf{u}_3$ is a linear tuple wrt $(u_{1,1}, u_{2,2}, g)$.

Group-Element Commitment. To commit a group element $X \in \mathbb{G}$, one chooses random coins $s_1, s_2, s_3 \in \mathbb{Z}_p$ and sets $\mathcal{C}(X) := (1, 1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} \odot \mathbf{u}_3^{s_3} = (u_{1,1}^{s_1} \cdot u_{3,1}^{s_3}, u_{2,2}^{s_2} \cdot u_{3,2}^{s_3}, X \cdot g^{s_1+s_2} \cdot u_{3,3}^{s_3})$.

- For a correct initialization of the commitment key ($\mathbf{u}_3$ is a linear tuple), we can re-write $\mathcal{C}(X)$ as $\mathcal{C}(X) = (u_{1,1}^a, u_{2,2}^b, X \cdot g^{a+b})$, for $a = s_1 + \lambda s_3$ and $b = s_2 + \mu s_3$. This is a linear encryption of X under the key $(u_{1,1}, u_{2,2})$ (see below). We thus have a perfectly binding commitment scheme, whereas it is hiding under the DLin assumption. A simulator that knows the discrete logarithms of $u_{1,1}$ and $u_{2,2}$ in basis g (the decryption key of the linear encryption) can extract X.
- In the case that $\mathbf{u}_3$ is a random tuple (random initialization), $\mathbf{u}_3 = (u_{3,1} = u_{1,1}^\lambda, u_{3,2} = u_{2,2}^\mu, u_{3,3} = g^\kappa)$, then we can re-write $\mathcal{C}(X)$ as $\mathcal{C}(X) = (u_{1,1}^a, u_{2,2}^b, X \cdot g^c)$, for $a = s_1 + \lambda s_3$, $b = s_2 + \mu s_3$, and $c = s_1 + s_2 + \kappa s_3$. It perfectly blinds X.

Scalar Commitment. To commit a scalar $x \in \mathbb{Z}_p$, one chooses random coins $\gamma_1, \gamma_2 \in \mathbb{Z}_p$ and sets $\mathcal{C}'(x) := (u_{3,1}^x, u_{3,2}^x, (u_{3,3}g)^x) \odot \mathbf{u}_1^{\gamma_1} \odot \mathbf{u}_3^{\gamma_2} = (u_{3,1}^{x+\gamma_2} \cdot u_{1,1}^{\gamma_1}, u_{3,2}^{x+\gamma_2}, u_{3,3}^{x+\gamma_2} \cdot g^{x+\gamma_1})$.

- For a correct initialization of the commitment key ($\mathbf{u}_3$ is a linear tuple), we can re-write $\mathcal{C}'(x)$ as $\mathcal{C}'(x) = (u_{1,1}^a, u_{2,2}^b, g^x \cdot g^{a+b})$, for $a = \lambda(x + \gamma_2) + \gamma_1$ and $b = \mu(x + \gamma_2)$. This is a linear encryption of g^x under the key $(u_{1,1}, u_{2,2})$, as above. A simulator that knows the discrete logarithms of $u_{1,1}$ and $u_{2,2}$ in basis g can extract g^x. In the case x is small (a bit), one can then extract x. Note that this commitment scheme is homomorphic: the product of $\mathcal{C}'(x)$ and $\mathcal{C}'(y)$ is a commitment of $x + y \bmod p$.
- In the case that $\mathbf{u}_3$ is a random tuple (random initialization), $\mathbf{u}_3 = (u_{3,1} = u_{1,1}^\lambda, u_{3,2} = u_{2,2}^\mu, u_{3,3} = g^\kappa)$, then we can re-write $\mathcal{C}'(x)$ as $\mathcal{C}'(x) = (u_{1,1}^a, u_{2,2}^b, g^x \cdot g^c)$, for $a = \lambda(x + \gamma_2) + \gamma_1$, $b = \mu(x + \gamma_2)$, and $c = \gamma_1 + \kappa(x + \gamma_2)$. It perfectly blinds x.

Proofs. Under the DLin assumption, the two initializations of the commitment key (linear tuple or random tuple) are indistinguishable. The former initialization provides a perfectly binding commitment and perfectly sound proofs, whereas the latter provides a perfectly hiding commitment and perfectly witness hiding proofs.

Relations are of three types, depending on the place of the commitments in the pairings. We thus use notation $\langle x \rangle$ for a committed scalar x (or $\langle X \rangle$ for a committed group element X). More details can be found in [GS08]:

- Multi-scalar multiplications equations: $\langle X^{r_1+r_2} \rangle = X^{\langle r_1 \rangle + \langle r_2 \rangle}$, to prove that the committed group element is equal to X raised to the sum of the committed scalars r_1 and r_2;
- Linear multi-scalar multiplications equations: $X_1^{\langle r_1 \rangle} = c_1$, to prove that the committed scalar r_1 is the one used in c_1;
- Quadratic equations: $\langle x \rangle(\langle x \rangle - 1) = 0$, to show that a committed scalar x is a bit.

In multi-scalar multiplication equations, the matrix containing the proof elements is composed of 9 group elements. Note that some optimizations can be made if we have linear equations. Quadratic equations require 6 elements only:

Assumption: DLin	$\mathbb{G}$
Commitments	3
Multi-scalar multiplication equations	9
- *Linear* multi-scalar multiplication equations	2
Quadratic equations	6

B.2 Waters Signatures

The Waters signature scheme is defined by the four algorithms.

- $\mathsf{Setup}(1^k)$: The scheme needs 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 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.
  We will sign messages $M = (M_1, \ldots, M_k) \in \{0,1\}^k$. To this aim, we need a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, and for convenience, we denote the *Waters' Hash* as $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. We also need an additional generator $h \xleftarrow{\$} \mathbb{G}$. The global parameters param consist of all these elements $(p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$.
- $\mathsf{SKeyGen}(\mathsf{param})$: Choose a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = X = g^x$, and the secret key as $\mathsf{sk} = Y = h^x$.
- $\mathsf{Sign}(\mathsf{sk} = Y, M; s)$: For some random $s \xleftarrow{\$} \mathbb{Z}_p$, define the signature as $\sigma = \big(\sigma_1 = Y \cdot \mathcal{F}(M)^s, \sigma_2 = g^{-s}\big)$.
- $\mathsf{Verif}(\mathsf{vk} = X, M, \sigma)$: One checks whether $e(g, \sigma_1) \cdot e(\mathcal{F}(M), \sigma_2) = e(X, h)$.

As shown by Waters [Wat05], this scheme is existentially unforgeable against chosen-message attacks (EUF-CMA) under the hardness of CDH. We note that signing and verifying can be performed without knowing the message M itself, it suffices to have $F = \mathcal{F}(M)$. Nevertheless, appropriate proofs of knowledge of M are required for the unforgeability security. We can thus replace M by the pair (F, Π_M) in both the signing and verifying algorithms. Since we are looking for randomizable signatures and encryption, we will use proofs in the Groth-Sahai framework, with a commitment C_M of M (bit-by-bit to make it extractable: $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k))$) and a proof that F is actually the evaluation of $\mathcal{F}$ on the committed M [FP09].

Theorem 11. *The Waters signature scheme is randomizable when we define, for valid signature σ and for random $s \xleftarrow{\$} \mathbb{Z}_p$*

- $\mathsf{Random}(\mathsf{vk}, (F, \Pi_M), \sigma = (\sigma_1, \sigma_2); s)$ *outputs* $\sigma' = (\sigma_1 \cdot F^s, \sigma_2 \cdot g^{-s})$.

Proof. It is clear that the following works:

$$\sigma = \mathsf{Sign}(\mathsf{sk}, (F, \Pi_M); s) \quad \Rightarrow \quad \mathsf{Random}(\mathsf{vk}, (F, \Pi_M), \sigma; s') = \mathsf{Sign}(\mathsf{sk}, (F, \Pi_M); s + s' \bmod p).$$

Due to the group structure of the random coins space, the re-randomization algorithm provides the correct distribution.

B.3 Linear Encryption

Linear encryption is defined by the four algorithms.

- $\mathsf{Setup}(1^k)$: The scheme needs 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 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. The global parameters param consist of these elements $(p, \mathbb{G}, \mathbb{G}_T, e, g)$.

– $\mathsf{EKeyGen}(\mathsf{param})$: Choose two random scalars $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, which define the secret key $\mathsf{dk} = (x_1, x_2)$, and the public key as $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$.
– $\mathsf{Encrypt}(\mathsf{pk} = (X_1, X_2), M; r_1, r_2)$: For a message $M \in \mathbb{G}$ and random scalars $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$, define the ciphertext as $c = \big(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot M\big)$.
– $\mathsf{Decrypt}(\mathsf{dk} = (x_1, x_2), c = (c_1, c_2, c_3))$: One computes $M = c_3/(c_1^{1/x_1} c_2^{1/x_2})$.

As shown by Boneh, Boyen and Shacham [BBS04], this scheme is semantically secure against chosen-plaintext attacks (IND-CPA) under the hardness of DLin.

Theorem 12. *The linear encryption scheme is randomizable when we define:*

– $\mathsf{Random}(\mathsf{pk}, M, c = (c_1, c_2, c_3); r'_1, r'_2)$ *outputs* $c' = (c_1 \cdot X_1^{r'_1}, c_2 \cdot X_2^{r'_2}, c_3 \cdot g^{r'_1+r'_2})$, *for random scalars* $r'_1, r'_2 \xleftarrow{\$} \mathbb{Z}_p$

Proof. It is clear that the following is a valid randomization:

$$c = (c_1, c_2, c_3) = \mathsf{Encrypt}(\mathsf{pk}, M; r_1, r_2)$$
$$\Rightarrow \mathsf{Random}(\mathsf{pk}, c; r'_1, r'_2) = \mathsf{Encrypt}(\mathsf{pk}, M; r_1 + r'_1 \bmod p, r_2 + r'_2 \bmod p) \ . \qquad \square$$

B.4 Signature on Encrypted Messages

We are now ready to generate a signature on an encryption of $\mathcal{F}(M)$. But as already explained, the signature scheme remains unforgeable on F with an additional proof of knowledge Π_M of M such that $F = \mathcal{F}(M)$.

– $\mathsf{Setup}(1^k)$: The scheme needs 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 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. For the signing part, we need an additional vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$, and a generator $h \xleftarrow{\$} \mathbb{G}$. The parameters param_e for the encryption part consist of $(p, \mathbb{G}, \mathbb{G}_T, e, g)$, whereas the parameters param_s for the signing part consist of $(p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$.
– $\mathsf{EKeyGen}(\mathsf{param}_e)$: Choose two random scalars $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, which define the secret key $\mathsf{dk} = (x_1, x_2)$, and the public key as $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$.
– $\mathsf{SKeyGen}(\mathsf{param}_s)$: Choose a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = X = g^x$, and the secret key as $\mathsf{sk} = Y = h^x$.
– $\mathsf{Encrypt}(\mathsf{pk} = (X_1, X_2), \mathsf{vk} = X, M; (r_1, r_2))$: For a message $M \in \{0,1\}^k$ and random scalars $r_1, r_2 \in \mathbb{Z}_p$, define the ciphertext as $c = \big(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \mathcal{F}(M)\big)$.
To guarantee our notion of unforgeability of signatures on ciphertexts, we also add some proofs of validity of the ciphertext:
  - A proof Π_r of knowledge of r_1 and r_2, used to encrypt $\mathcal{F}(M)$, which consists of bit-by-bit commitments $C_1 = (\mathcal{C}'(r_{1,1}), \ldots, \mathcal{C}'(r_{1,\ell}))$ and $C_2 = (\mathcal{C}'(r_{2,1}), \ldots, \mathcal{C}'(r_{2,\ell}))$, where ℓ is the bit-length of the order p, and proofs that each sub-commitment is indeed a bit commitment: as explained above, each sub-commitment with the quadratic-equation proof of bit requires 6 group elements. One then has to prove that c_1 and c_2 are well-formed w.r.t. these commitments: $c_1 = X_1^{\langle r_1 \rangle}$ and $c_2 = X_2^{\langle r_2 \rangle}$, where $\mathcal{C}'(r_i) = \mathcal{C}'(\sum_{j \in \{1,\ldots,\ell\}} 2^{j-1} r_{i,j}) = \prod_{j \in \{1,\ldots,\ell\}} \mathcal{C}'(r_{i,j})^{2^{j-1}}$. These linear multi-scalar multiplication equation proofs require 2 group elements each. Globally, Π_r is composed of $18\ell + 4$ group elements.
  - A proof Π_M of knowledge of M in c, the encrypted $\mathcal{F}(M)$, which consists of a bit-by-bit commitment $C_M = (\mathcal{C}'(M_1), \ldots, \mathcal{C}'(M_k))$, with proofs that each commitment is also indeed a bit commitment: $6k$ group elements. One then has to prove that c_3 is well-formed: $c_3 = (u_0 \prod_{i \in \{1,\ldots,k\}} u_i^{\langle M_i \rangle}) \cdot g^{\langle r_1 \rangle + \langle r_2 \rangle}$, which is a linear multi-scalar multiplication equation proof: 2 additional group elements. Therefore Π_M is composed of $9k + 2$ group elements.

As a consequence, the global proof, that we denote by Π for the sake of clarity, can be done with randomizable commitments and proofs, using the Groth-Sahai methodology [GS08, FP09], and consists of $9k + 18\ell + 6$ group elements.

- $\mathsf{Sign}(\mathsf{sk} = Y, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi); s)$: When one wants to sign a ciphertext $c = (c_1, c_2, c_3)$, one first checks if the latter is valid, using Π, and produces $\sigma = (c_1^s,\ c_2^s,\ Y \cdot c_3^s;\ X_1^s,\ X_2^s, g^s)$. The second part (X_1^s, X_2^s, g^s) will enable randomization.
- $\mathsf{Decrypt}(\mathsf{dk} = (x_1, x_2), \mathsf{vk} = X, (c = (c_1, c_2, c_3), \Pi))$: On a valid ciphertext (one can check with Π), if one knows the decryption key $\mathsf{dk} = (x_1, x_2)$, one can get back $F = \mathcal{F}(M)$, and one can also complete the proof Π into a proof of knowledge of M: $F = c_3/(c_1^{1/x_1} c_2^{1/x_2})$.
- $\mathsf{Verif}(\mathsf{vk} = X, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi), \sigma = (\sigma_1, \sigma_2, \sigma_3; \sigma_4, \sigma_5, \sigma_6))$: In order to check the validity of the signature, one first checks whether the proof on the ciphertext is valid and whether the following pairing equations are verified: $e(\sigma_3, g) = e(g, \mathsf{vk}) \cdot e(c_3, \sigma_6)$ and

$$e(\sigma_1, X_1) = e(c_1, \sigma_4) \qquad e(\sigma_2, X_2) = e(c_2, \sigma_5) \qquad e(\sigma_1, g) = e(c_1, \sigma_6) \qquad e(\sigma_2, g) = e(c_2, \sigma_6)$$

Theorem 13. *The Waters signature on linear ciphertexts is extractable when one defines*

- $\mathsf{SigExt}(\mathsf{dk} = (x_1, x_2), \mathsf{vk} = X, \sigma)$*: On a valid signature, if one knows the decryption key* $\mathsf{dk} = (x_1, x_2)$*), one can get back a signature on* M *(or* $F = \mathcal{F}(M)$*):* $\Sigma = (\Sigma_1 = \sigma_3/(\sigma_1^{1/x_1}\sigma_2^{1/x_2}), \Sigma_2 = \sigma_6^{-1})$*. Note that one can also get the same value from the coins for encryption* r_1, r_2*:* $\Sigma = (\Sigma_1 = \sigma_3/\sigma_6^{r_1+r_2}, \Sigma_2 = \sigma_6^{-1})$*.*

Proof. The proof follows from the fact that

$$\Sigma_1 = \sigma_3/(\sigma_1^{1/x_1}\sigma_2^{1/x_2}) = Y \cdot c_3^s/(c_1^{s/x_1} c_2^{s/x_2}) = Y \cdot g^{s(r_1+r_2)} \cdot \mathcal{F}(M)^s / g^{sr_1} g^{sr_2} = Y \cdot \mathcal{F}(M)^s,$$
$$\Sigma_2 = \sigma_6^{-1} = g^{-s}.$$

□

Theorem 14. *The Waters signature on linear ciphertexts is randomizable when one defines*

- $\mathsf{Random}(\mathsf{vk} = X, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi), \sigma = (\sigma_1, \sigma_2, \sigma_3; \sigma_4, \sigma_5, \sigma_6); r_1', r_2', s')$*: In order to randomize the signature and the ciphertext, the algorithm outputs:*

$$c' = (c_1 \cdot X_1^{r_1'}, c_2 \cdot X_2^{r_2'}, c_3 \cdot g^{r_1'+r_2'})$$
$$\sigma' = (\sigma_1 \cdot c_1^{s'} \times \sigma_4^{r_1'} \cdot X_1^{r_1's'}, \sigma_2 \cdot c_2^{s'} \times \sigma_5^{r_2'} \cdot X_2^{r_2's'}, \sigma_3 \cdot c_3^{s'} \times \sigma_6^{r_1'+r_2'} \cdot g^{(r_1'+r_2')s'}; \sigma_4 \cdot X_1^{s'}, \sigma_5 \cdot X_2^{s'}, \sigma_6 \cdot g^{s'})$$

together with a randomization Π' *of* Π*.*

Proof. On the input $(\mathsf{vk} = X, \mathsf{pk} = (X_1, X_2), (c = (c_1, c_2, c_3), \Pi), \sigma = (\sigma_1, \sigma_2, \sigma_3; \sigma_4, \sigma_5, \sigma_6); r_1', r_2', s')$, where for some random scalars $s, r_1, r_2 \in \mathbb{Z}_p$,

$$c = (c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \mathcal{F}(M))$$
$$\sigma = (c_1^s = X_1^{r_1 s}, c_2^s = X_2^{r_2 s}, Y \cdot c_3^s = Y \cdot g^{(r_1+r_2)s} \cdot \mathcal{F}(M)^s; X_1^s, X_2^s, g^s)$$

the algorithm outputs (where $R_1 = r_1 + r'_1$, $R_2 = r_2 + r'_2$ and $S = s + s'$):

$$\begin{aligned}
c' &= (c_1 \cdot X_1^{r'_1} = X_1^{r_1+r'_1}, c_2 \cdot X_2^{r'_2} = X_2^{r_2+r'_2}, c_3 \cdot g^{r'_1+r'_2} = g^{r_1+r'_1+r_2+r'_2} \cdot \mathcal{F}(M)) \\
&= (X_1^{R_1}, X_2^{R_2}, g^{R_1+R_2} \cdot \mathcal{F}(M)) \\
\sigma' &= (\sigma_1 \cdot c_1^{s'} \times \sigma_4^{r'_1} \cdot X_1^{r'_1 s'}, \sigma_2 \cdot c_2^{s'} \times \sigma_5^{r'_2} \cdot X_2^{r'_2 s'}, \sigma_3 \cdot c_3^{s'} \times \sigma_6^{r'_1+r'_2} \cdot g^{(r'_1+r'_2)s'}; \sigma_4 \cdot X_1^{s'}, \sigma_5 \cdot X_2^{s'}, \sigma_6 \cdot g^{s'}) \\
&= (X_1^{r_1 s} \cdot X_1^{r_1 s'} \times X_1^{r'_1 s} \cdot X_1^{r'_1 s'}, X_2^{r_2 s} \cdot X_2^{r_2 s'} \times X_2^{r'_2 s} \cdot X_2^{r'_2 s'}, \\
&\qquad Y \cdot \mathcal{F}(M)^s \cdot g^{(r_1+r_2)s} \cdot \mathcal{F}(M)^{s'} \cdot g^{(r_1+r_2)s'} \times g^{(r'_1+r'_2)s} \cdot g^{(r'_1+r'_2)s'}; X_1^{s+s'}, X_2^{s+s'}, g^{s+s'}) \\
&= (X_1^{(r_1+r'_1)(s+s')}, X_2^{(r_2+r'_2)(s+s')}, Y \cdot \mathcal{F}(M)^{s+s'} \cdot g^{(r_1+r_2+r'_1+r'_2)(s+s')}; X_1^{s+s'}, X_2^{s+s'}, g^{s+s'}) \\
&= (X_1^{R_1 S}, X_2^{R_2 S}, Y \cdot \mathcal{F}(M)^S \cdot g^{(R_1+R_2)S}; X_1^S, X_2^S, g^S) = ({c'_1}^S, {c'_2}^S, Y \cdot {c'_3}^S; X_1^S, X_2^S, g^S).
\end{aligned}$$

We thus have

$$\mathsf{Random}(\mathsf{vk}, \mathsf{pk}, c, \sigma; r'_1, r'_2, s') = \mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c'; s + s'), \text{ where } c' = \mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, M; r_1 + r'_1, r_2 + r'_2).$$

□

Theorem 15. *The Waters signature on linear ciphertexts is unforgeable (in the UF sense) under the* CDH *assumption in* $\mathbb{G}$.

Proof. Let us denote $\mathcal{SC}$ our above signature on ciphertexts (but omit it in the subscripts for clarity), and $\mathcal{S}$ the Waters signature scheme. We know that the latter is existentially unforgeable under the CDH assumption. Let us assume that $\mathcal{A}$ is able to break the unforgeability of $\mathcal{SC}$. We will build an adversary $\mathcal{B}$ against that Waters signature scheme. We note that $\mathcal{B}$ generated the parameters for the commitments for the proof Π of knowledge of M, r_1 and r_2, so that it can extract the values.

- $\mathsf{Setup}(1^k)$: we first run the $\mathsf{Setup}_{\mathcal{S}}(1^k)$ algorithm, from which we get $\mathsf{param}_{\mathcal{S}} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$. We set $\mathsf{param}_s = \mathsf{param}_{\mathcal{S}} = (p, \mathbb{G}, \mathbb{G}_T, e, g, h, \boldsymbol{u})$, and $\mathsf{param}_e = (p, \mathbb{G}, \mathbb{G}_T, e, g)$. $\mathcal{B}$ sets the commitment parameters so that it can extract committed values.
- $\mathsf{EKeyGen}(\mathsf{param}_e)$: for each new key request, $\mathcal{B}$ chooses two random scalars $x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p$, which define the secret key $\mathsf{dk} = (x_1, x_2)$, and the public key as $\mathsf{pk} = (X_1 = g^{x_1}, X_2 = g^{x_2})$.
- $\mathsf{SKeyGen}(\mathsf{param}_s)$: for the unique signing key request, one gets the verification key $\mathsf{vk}_{\mathcal{S}}$ from the Waters EUF-CMA security game. $\mathcal{B}$ sets $\mathsf{vk} = \mathsf{vk}_{\mathcal{S}}$.
- $\mathcal{A}$ can now access a signing oracle, with queries of the form $\mathsf{Sign}(\mathsf{vk}, \mathsf{pk}, \cdot)$, for any pk and ciphertext of its choice. But the ciphertext looks like $c = (c_1, c_2, c_3)$ together with $\Pi = (\Pi_M, \Pi_r)$.
  - If the tuple (c, Π) is not valid, then $\mathcal{B}$ returns $\perp$;
  - Otherwise, $\mathcal{B}$ can extract M from the proof of knowledge Π_M, that contains a bit-by-bit extractable commitment C_M of M. It then queries $\mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}_{\mathcal{S}}, M)$ to the signing oracle, and adds (vk, M) to the SM set. It receives back $\sigma' = (\sigma'_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma'_2 = g^{-s})$. $\mathcal{B}$ can also extract r_1 and r_2 from the bit-by-bit commitments C_1 and C_2 included in the proof Π_r of knowledge of r_1 and r_2. It then returns the signature σ defined as

$$\begin{pmatrix} \sigma_1 = {\sigma'_2}^{-r_1 x_1} = g^{s r_1 x_1} = X_1^{s r_1}, \sigma_2 = {\sigma'_2}^{-r_2 x_2} = X_2^{s r_2}, & \sigma_3 = \sigma'_1 {\sigma'_2}^{-r_1 - r_2} = \mathsf{sk} \cdot \mathcal{F}(M)^s \cdot g^{s(r_1+r_2)}, \\ \sigma_4 = {\sigma'_2}^{-x_1} = g^{s x_1} = X_1^s, & \sigma_5 = {\sigma'_2}^{-x_2} = g^{s x_2} = X_2^s, \sigma_6 = {\sigma'_2}^{-1} = g^s \end{pmatrix}$$

- After a polynomial number of queries, $\mathcal{A}$ outputs, with non-negligible probability, a valid signature σ on a valid ciphertext (c, Π). As above, one can extract the message M, and for a valid forgery, one needs $M \neq \perp$ and $(M, \mathsf{vk}) \notin \mathsf{SM}$. Using SigExt, as shown above, one thus gets a valid Waters signature on M: $\sigma_{\mathcal{B}} = (\sigma_3 / (\sigma_1^{1/x_1} \sigma_2^{1/x_2}) = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_6^{-1} = g^{-s})$. This breaks the EUF-CMA property of the Waters signature scheme, that holds under the CDH assumption.

□

C Asymmetric Waters Signature on ElGamal Ciphertexts

As symmetric bilinear groups are in general less efficient than *asymmetric* groups, we show how to instantiate our primitive with ElGamal encryption in an asymmetric pairing setting, relying on the SXDH assumption. As for the previous DLin-based scheme, we need to modify the Waters signature to make the resulting scheme more efficient. But first, we have to adapt the Waters signature to the asymmetric setting.

C.1 Assumptions

The security of Waters signatures in asymmetric bilinear groups can be proved 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 16 (The Advanced Computational Diffie-Hellman problem (CDH^+)). *Let us be given two (multiplicative) groups $(\mathbb{G}_1, \mathbb{G}_2)$ 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; since we will use it in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$, we assume SXDH, defined below. In the asymmetric setting, the ElGamal requires the DDH assumption:

Definition 17 (Decisional Diffie-Hellman Assumption (DDH)). *Let $\mathbb{G}$ be a cyclic group of prime order p. The DDH assumption states that given $(g, g^a, g^b, g^c) \in \mathbb{G}$, it is hard to determine whether $c = ab$.*

Definition 18 (Symmetric external Diffie-Hellman Assumption (SXDH) [BBS04]). *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 the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

C.2 Groth-Sahai Commitments

As above, several elements (group elements or scalars) will have to be committed so that proofs can be done on them. We will use SXDH-based Groth-Sahai commitments: they require a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, 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

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

Initialization of the parameters should be: $\boldsymbol{u_1} = (g_1, u)$ 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 $\boldsymbol{u_2} = (g_1^\mu, g_1^{\lambda\mu})$. In the hiding setting, 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 it is the same in $\mathbb{G}_2$ for $\boldsymbol{v}$.

Group Element Commitment. To commit to $X \in \mathbb{G}_1$, one chooses randomness $s_1, s_2 \in \mathbb{Z}_p$ and sets $\mathcal{C}(X) = (1, X) \odot \mathbf{u}_1^{s_1} \odot \mathbf{u}_2^{s_2} = (u_{1,1}^{s_1} \cdot u_{2,1}^{s_2}, X \cdot u_{1,2}^{s_1} \cdot u_{2,2}^{s_2})$.

- For a correct initialization of the commitment key, we can re-write $\mathcal{C}(X)$ as $\mathcal{C}(X) = (g_1^a, X \cdot u^a)$, for $a = s_1 + \mu s_2$. This is an ElGamal encryption of X under the key $(u_{1,1} = g_1, u_{1,2} = u = g_1^\lambda)$ (see below). A simulator that knows the discrete logarithm λ of u in basis g_1 can extract X.
- In the case that $\boldsymbol{u}_2 = \boldsymbol{u_1}^\mu \odot (1, g_1)^{-1} = (u_{1,1}^\mu, u_{1,2}^\mu / g_1)$, we can re-write $\mathcal{C}(X)$ as $\mathcal{C}(X) = (g_1^a, X/g_1^b \cdot u^a)$, for $a = s_1 + \mu s_2$, $b = s_1$, two random independent values: this is an encryption of X/g_1^b, for a random b. It thus perfectly blinds X.

The commitment in $\mathbb{G}_2$ follows the same rules, with $\boldsymbol{v}$ and g_2 instead of $\boldsymbol{u}$ and g_1.

Scalar Commitment. To commit to $x \in \mathbb{Z}_p$ in $\mathbb{G}_1$, one chooses $\gamma \in \mathbb{Z}_p$ and sets $\mathcal{C}'(x) = (u_{2,1}^x, (u_{2,2}g_1)^x) \odot \mathbf{u}_1^\gamma = (u_{1,1}^\gamma u_{2,1}^x, u_{1,2}^\gamma (u_{2,2}g_1)^x)$.

- For a correct initialization of the commitment key, we can re-write $\mathcal{C}'_1(X)$ as $\mathcal{C}'_1(x) = (g_1^a, g_1^x \cdot u^a)$, for $a = \gamma + \mu x$. This is an ElGamal encryption of g_1^x under the key $(u_{1,1} = g_1, u_{1,2} = u = g_1^\lambda)$ (see below). A simulator that knows the discrete logarithm λ of u in basis g_1 can extract g_1^x. In the case x is small (a bit), one can then extract x. Note that this commitment scheme is homomorphic: the product of $\mathcal{C}'_1(x)$ and $\mathcal{C}'_1(y)$ is a commitment of $x + y \bmod p$.
- In the case that $\boldsymbol{u}_2 = \boldsymbol{u}_1{}^\mu \odot (1, g_1)^{-1} = (u_{1,1}^\mu, u_{1,2}^\mu / g_1)$, we can re-write $\mathcal{C}'_1(X)$ as $\mathcal{C}'_1(X) = (u_{1,1}^{\gamma+\mu x}, u_{1,2}^{\gamma+\mu x}) = (g_1^a, u^a)$, for $a = \gamma + \mu x$. It perfectly blinds x.

The same things can be done with $\mathcal{C}'_2$, $\boldsymbol{v}$ in $\mathbb{G}_2$

Proofs. Under the SXDH assumption, the two initializations of the commitment key (perfectly binding or perfectly hiding) are indistinguishable. The former provides perfectly sound proofs, whereas the latter provides perfectly witness hiding proofs. A Groth-Sahai proof, is a pair of elements $(\pi, \theta) \in \mathbb{G}_1^{2\times 2} \times \mathbb{G}_2^{2\times 2}$. These elements are constructed to help verifying pairing relations on committed values. Being able to produce a valid pair implies knowing plaintexts verifying the appropriate relation. As above, we will note $\langle x \rangle_1$ for a committed scalar x in $\mathbb{G}_1$, $\langle x \rangle_2$ for a committed scalar x in $\mathbb{G}_2$,(or $\langle X \rangle$ for a committed group element X). We will use:

- Linear Pairing Product equations: $e(\langle X_1^r \rangle, g_2) = e(g_1^{\langle r \rangle_1}, X_2)$, to prove that the group element commit ed, is the public key raised to the value of the committed scalar in $\mathbb{G}_1$.
- Multi-scalar multiplications equations in $\mathbb{G}_1$: $\langle X_1^r \rangle = X_1^{\langle r \rangle_2}$, to prove the group element, committed in $\mathbb{G}_1$, is equal to X_1 raised to the value of the scalar committed in $\mathbb{G}_2$.
- Linear Multi-scalar multiplications equations in $\mathbb{G}_1$: $c = U_1^{\langle r \rangle_2}$, to prove the group element c (in $\mathbb{G}_1$), equal to U_1 raised to the committed scalar r.
- Quadratic equations: $\langle x \rangle_1 (\langle x' \rangle_2 - 1) = 0$, to show that a committed scalar x is truly a bit $\in \{0, 1\}$. (This equation with its symmetric are enough to prove that if x is committed in $\mathbb{G}_1$ and x' in $\mathbb{G}_2$ we have : $x = x'$ and $x \in \{0, 1\}$

Multi-scalar multiplication equations will require group elements (2 in $\mathbb{G}_1$, 4 in $\mathbb{G}_2$), whereas quadratic equations will only require two in each. Once again, some additional optimizations can be made if we have linear equations:

Assumption: SXDH	$\mathbb{G}_1$	$\mathbb{G}_2$
Commitments	2 or 2	
Linear Pairing Product equations	0	2
Multi-scalar multiplication equations in $\mathbb{G}_1$	2	4
- *Linear* Multi-scalar multiplication equations in $\mathbb{G}_1$	1	0
Quadratic equations	2	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 so require to use 2 quadratic equations by bit.

C.3 Asymmetric Waters Signature Scheme

First, we present a result that is of independent interest: the asymmetric Waters signature.

- $\mathsf{Setup}(1^k)$: The scheme needs 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 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. We will sign messages $M = (M_1, \ldots, M_k) \in \{0, 1\}^k$. To this aim, we need a vector

$\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, and for convenience, we denote the *Waters Hash* as $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. We also need an additional generator $h_1 \xleftarrow{\$} \mathbb{G}_1$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \boldsymbol{u})$.

- $\mathsf{SKeyGen}(\mathsf{param})$: Choose a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (X_1, X_2) = (g_1^x, g_2^x)$, and the secret key as $\mathsf{sk} = Y = h_1^x$.
- $\mathsf{Sign}(\mathsf{sk} = Y, M; s)$: For some random $s \xleftarrow{\$} \mathbb{Z}_p$, define the signature as $\sigma = \big(\sigma_1 = Y(\mathcal{F}(M))^s, \sigma_2 = g_1^{-s}, \sigma_3 = g_2^{-s}\big)$.
- $\mathsf{Verif}(\mathsf{vk} = (X_1, X_2), M, \sigma)$: One checks whether $e(\sigma_1, g_2) \cdot e(\mathcal{F}(M), \sigma_3) = e(h_1, X_2)$, and $e(\sigma_2, g_2) = e(g_1, \sigma_3)$.

As for the original scheme, signing and verifying can be performed when $F = \mathcal{F}(M)$ is given instead of M. For both algorithms, we can replace M by $(\mathcal{F}(M), \Pi_M)$. In addition, to verify a signature, only the second component X_2 of the public key is used. We could define a scheme with public keys $\mathsf{vk} = X_2$ and prove it secure under a slightly weaker assumption. However, we defined the scheme with keys of the form (X_1, X_2) since we require it for signatures on encrypted messages (see the next section).

Theorem 19. *The Asymmetric Waters signature scheme is randomizable and existentially unforgeable under the* CDH^+ *assumption.*

Proof. First, let us define $\mathsf{Random}(\mathsf{vk}, (F, \Pi_M), \sigma = (\sigma_1, \sigma_2, \sigma_3); s')$ to output $\sigma' = (\sigma_1 \cdot F^{s'}, \sigma_2 \cdot g_1^{-s'}, \sigma_3 \cdot g_2^{-s'})$, for random $s' \xleftarrow{\$} \mathbb{Z}_p$. We easily see it corresponds to $\mathsf{Sign}(\mathsf{sk}, F, \Pi_M; s + s' \bmod p)$. Because of the group structure of $\mathbb{Z}_p$, we get appropriate distributions.

Let $\mathcal{A}$ be an adversary breaking the existential unforgeability of the above signature scheme, i.e. after at most q_s signing queries, it succeeds in building a new signature with probability ϵ. Let $(g_1, g_2, \lambda = (g_1^a, g_2^a), \mu = g_1^b)$ be an CDH^+-instance. We show how an adversary $\mathcal{B}$ can compute g_1^{ab} thanks to $\mathcal{A}$.

$\mathsf{Setup}_{\mathcal{S}}$. Pick a random position $j \xleftarrow{\$} \{0, \ldots, k\}$, choose random indices $y_0, y_1, \ldots, y_k \xleftarrow{\$} \{0, \ldots, 2q_s - 1\}$, and random scalars $z_0, z_1, \ldots, z_k \xleftarrow{\$} \mathbb{Z}_p$. One defines $X_1 = g_1^a$, $X_2 = g_2^a$, $h_1 = g_1^b$, $u_0 = h_1^{y_0 - 2jq_s} g^{z_0}$, $u_i = h_1^{y_i} g_1^{z_i}$.

Signing queries. To answer a signing query on m, with a message $M = (M_i)$, we define

$$H = -2jq_s + y_0 + \sum_i y_i M_i, \quad J = z_0 + \sum_i z_i M_i : \mathcal{F}(M) = h_1^H g_1^J.$$

If $H \equiv 0 \pmod p$ then abort, otherwise set $\sigma = (X_1^{-J/H} \mathcal{F}(M)^s, X_1^{1/H} g_1^{-s}, X_2^{1/H} g_2^{-s})$. Defining $\tilde{s} = s - a/H$, we have:

$$\begin{aligned}
\sigma &:= \big(X_1^{-J/H} (h_1^H g_1^J)^s,\, X_1^{1/H} g_1^{-s},\, X_2^{1/H} g_2^{-s}\big) \\
&= \big(X_1^{-J/H} (h_1^a g_1^{Ja/H}) (\mathcal{F}(M))^{\tilde{s}},\, X_1^{1/H} g_1^{-a/H} g_1^{-\tilde{s}},\, X_2^{1/H} g_2^{-a/H} g_2^{-\tilde{s}}\big) \\
&= \big(h_1^a (\mathcal{F}(M))^{\tilde{s}},\, g_1^{-\tilde{s}},\, g_2^{-\tilde{s}}\big).
\end{aligned}$$

After at most q_s signing queries $\mathcal{A}$ outputs a forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*)$ on M^*. As before, we define

$$H^* = -2jq_s + y_0 + \sum_i y_i M_i^*, \quad J^* = z_0 + \sum_i z_i M_i^* : \mathcal{F}(M^*) = h_1^{H^*} g_1^{J^*}.$$

If $H^* \not\equiv 0 \pmod p$ then abort, otherwise, as shown above, for some s^*, $\sigma^* = (h_1^a \mathcal{F}(M^*)^{s^*}, g_1^{-s^*}, g_2^{-s^*})$, and thus $\sigma^* = (h_1^a g_1^{J^* s^*}, g_1^{-s^*}, g_2^{-s^*})$. As a consequence, $\sigma_1^* (\sigma_2^*)^{J^*} = h_1^a = g_1^{ab}$: one has solved the CDH^+ problem. As above, the latter case occurs with good probability. ◻

C.4 ElGamal Encryption

ElGamal encryption is defined by the four algorithms.

- $\mathsf{Setup}(1^k)$: The scheme needs a pairing-friendly system $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, with respective generators g_1, g_2 and $g_t = e(g_1, g_2)$.
- $\mathsf{EKeyGen}(\mathsf{param})$: One chooses a scalar α which defines $U_1 = g_1^\alpha$. (We can do the same in $\mathbb{G}_2$ with a scalar β which defines $U_2 = g_2^\beta$).
- $\mathsf{Encrypt}(\mathsf{pk} = u_1, M; r)$: The algorithm is given a random $r \in \mathbb{Z}_p$ and publishes $c = (c_1 = \mathcal{F}(M) \cdot U_1^r, c_2 = g_1^r)$.
- $\mathsf{Decrypt}(\mathsf{dk} = \alpha, c = (c_1, c_2))$: One computes $\mathcal{F}(M) = c_1 / c_2^\alpha$.

This scheme is semantically secure against chosen-plaintext attacks (IND-CPA) under the hardness of DDH in the appropriate group.

Theorem 20. *The ElGamal encryption scheme is randomizable when we define*

- $\mathsf{Random}(\mathsf{pk}, M, c = (c_1, c_2); r')$ *outputs* $c' = (c_1 \cdot U_1^{r'}, c_2 \cdot g_1^{r'})$, *for a random scalar* $r' \xleftarrow{\$} \mathbb{Z}_p$

Proof. It is clear that this is a valid re-randomization, due to the group structure of $\mathbb{Z}_p$:

$$c = \mathsf{Encrypt}(\mathsf{pk}, M; r) \Rightarrow \mathsf{Random}(\mathsf{pk}, c; r') = \mathsf{Encrypt}(\mathsf{pk}, M; r + r').$$

□

C.5 Signature on Encrypted Messages

We now want to generate a signature on an encryption of $\mathcal{F}(M)$. But as already explained, the signature scheme remains unforgeable on F with an additional proof of knowledge Π_M of M such that $F = \mathcal{F}(M)$.

- $\mathsf{Setup}(1^k)$: The system generates a pairing-friendly system $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$, with respective generators g_1, g_2 and $g_t = e(g_1, g_2)$. For the signing part, we need an additional vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, and a generator $h_1 \xleftarrow{\$} \mathbb{G}_1$.
- $\mathsf{EKeyGen}(\mathsf{param}_e)$: Choose a random scalar $\alpha \xleftarrow{\$} \mathbb{Z}_p$, which defines the secret key $\mathsf{dk} = \alpha$, and the public key as $\mathsf{pk} = U_1 = g_1^\alpha$.
- $\mathsf{SKeyGen}(\mathsf{param}_s)$: Choose a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (X_1 = g_1^x, X_2 = g_2^x)$, and the secret key as $\mathsf{sk} = Y = h_1^x$.
- $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, M; r)$: For some message $M \in \{0,1\}^k$ and some random scalar $r \in \mathbb{Z}_p$, define the ciphertext as $c = (\mathcal{F}(M) \cdot U_1^r, g_1^r)$.
  For making the security proof work, we also add some proofs of validity of the ciphertext:
  - A proof Π_r of knowledge of r, used to encrypt $\mathcal{F}(M)$, which consists of the bit-commitments in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$, $C_1 = (\mathcal{C}'_1(r_1), \ldots, \mathcal{C}'_1(r_\ell), \mathcal{C}'_2(r_1), \ldots, \mathcal{C}'_2(r_{1,\ell}))$, where ℓ is the bit-length of the order p, and proofs that each sub-commitments if indeed a bit commitment: they correspond to two quadratic equations, and consist each of 2 group elements in $\mathbb{G}_1$ and 2 group elements in $\mathbb{G}_2$ for each bit. One then has to prove that c_2 is well-formed w.r.t. these commitments, $c_2 = U_1^{\langle r \rangle_2}$, where the commitment of r can be obtained as above granted the homomorphic property. This equation is a linear multi-scalar multiplication equation in $\mathbb{G}_1$, requiring 1 group element in $\mathbb{G}_1$. Therefore Π_r is composed of $6\ell + 1$ group elements in $\mathbb{G}_1$ and 6ℓ in $\mathbb{G}_2$.

- A proof Π_M of knowledge of M in c, the encrypted $\mathcal{F}(M)$, which consists of the bit-commitments in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$, $C_M = (\mathcal{C}'_1(M_1), \ldots, \mathcal{C}'_1(M_k), \mathcal{C}'_2(M_1), \ldots, \mathcal{C}'_2(M_k))$, with proofs that each commitment is also indeed a bit commitment: $6k$ group elements in $\mathbb{G}_1$ and $6k$ in $\mathbb{G}_2$. One then has to prove that c_1 is well-formed: $c_1 = (u_0 \prod_{i\in[1,k]} u_i^{\langle M_i\rangle_2}) \cdot U_1^{\langle r\rangle_2}$, which is a linear multi-scalar multiplication equation in $\mathbb{G}_1$, and so only needs 1 extra group element in $\mathbb{G}_1$. Therefore Π_M is composed of $6k+1$ elements in $\mathbb{G}_1$ and $6k$ group elements in $\mathbb{G}_2$.

  As above, for the sake of clarity, we denote by Π the global additional proof, which consists of $6k+6\ell+2$ group elements in $\mathbb{G}_1$ and $6k+6\ell$ group elements in $\mathbb{G}_2$.
- $\mathsf{Sign}(\mathsf{sk} = Y, \mathsf{pk} = U_1, (c = (c_1, c_2), \Pi); s)$: When one wants to sign a ciphertext $c = (c_1, c_2)$, one first checks if the latter is valid using Π and produces: $\sigma = (Y \cdot c_1^s, c_2^s, U_1^s, g_1^s, g_2^s)$ if it is valid or $\perp$ in the other case.
- $\mathsf{Random}(\mathsf{vk} = (X_1, X_2), \mathsf{pk} = U_1, (c = (c_1, c_2), \Pi), \sigma; r', s')$: This algorithm is given random scalars $r', s' \in \mathbb{Z}_p$ and publishes

$$c' = (c'_1 = c_1 \cdot U_1^{r'}, c'_2 = c_2 \cdot g_1^{r'})$$
$$\sigma' = (\sigma'_1 = \sigma_1 \cdot c_1^{s'} \cdot \sigma_3^{r'} \cdot U_1^{s'r'}, \sigma'_2 = \sigma_2 \cdot c_2^{s'} \cdot \sigma_4^{r'} \cdot g_1^{r's'}, \sigma'_3 = \sigma_3 \cdot U_1^{s'}, \sigma'_4 = \sigma_4 \cdot g_1^{s'}, \sigma'_5 = \sigma_5 \cdot g_2^{s'}),$$

  together with a randomization Π' of Π.
- $\mathsf{Verif}(\mathsf{vk} = (X_1, X_2), \mathsf{pk} = U_1, c, \sigma)$: In order to check the validity of the signature, one checks if Π is valid and if the following pairing equations are verified:

$$e(\sigma_2, g_2) = e(c_2, \sigma_5) \quad e(\sigma_3, g_2) = e(U_1, \sigma_5) \quad e(\sigma_4, g_2) = e(g_1, \sigma_5) \quad e(\sigma_1, g_2) = e(h_1, X_2) \cdot (c_1, \sigma_5).$$

- $\mathsf{SigExt}(\mathsf{dk}, \mathsf{vk}, \sigma)$: On a valid signature, if one knows the decryption key $\mathsf{dk} = \alpha$, one can get back a signature on M (of $F = \mathcal{F}(M)$):

$$\Sigma = (\Sigma_1 = \sigma_1/\sigma_2^\alpha, \Sigma_2 = \sigma_4^{-1}, \Sigma_3 = \sigma_4^{-1}).$$

  Note that one can also get the same value from the encryption random coins r, since $\Sigma_1 = \sigma_1/U_1^r$.

Theorem 21. *The Asymmetric Waters signature on ElGamal ciphertexts is randomizable.*

Proof. Let's show that $\mathsf{Random}(\mathsf{vk}, \mathsf{pk}, c, \sigma; r', s')$ is a valid randomization. We have:

$$\sigma'_1 = \sigma_1 \cdot c_1^{s'} \cdot \sigma_3^{r'} \cdot U_1^{s'r'} = (Y \cdot \mathcal{F}(M)^s \cdot U_1^{sr}) \cdot (\mathcal{F}(M)^{s'} \cdot U_1^{s'r}) \cdot U_1^{sr'} \cdot U_1^{s'r'} = Y \cdot \mathcal{F}(M)^{s+s'} \cdot U_1^{(s+s')(r+r')}$$
$$\sigma'_2 = \sigma_2 \cdot c_2^{s'} \cdot \sigma_4^{r'} \cdot g_1^{s'r'} = g_1^{sr} \cdot g_1^{s'r} \cdot g_1^{sr'} \cdot g_1^{s'r'} = g_1^{(s+s')(r+r')}$$
$$\sigma'_3 = \sigma_3 \cdot U_1^{s'} = U_1^{s+s'} \qquad \sigma'_4 = \sigma_4 \cdot g_1^{s'} = g_1^{s+s'} \qquad \sigma'_5 = \sigma_5 \cdot g_2^{s'} = g_2^{s+s'}.$$

This is straightforward for the ciphertext, so:

$$\mathsf{Random}(\mathsf{vk}, \mathsf{pk}, c, \sigma; r', s') = \mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c'; s + s') \text{ where } c' = \mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, M; r + r').$$

The group structure of $\mathbb{Z}_p$ leads to the conclusion. □

Theorem 22. *The Asymmetric Waters signature on ElGamal ciphertexts is unforgeable (in the UF sense) under the* CDH^+ *assumption.*

Proof. Let us denote $\mathcal{SC}$ our above signature on ciphertexts (but omit it in the subscripts for clarity), and $\mathcal{S}$ the asymmetric Waters signature scheme. We know that the latter is existentially unforgeable under the CDH^+ assumption. Let us assume that $\mathcal{A}$ is able to break the unforgeability of $\mathcal{SC}$. We will build an adversary $\mathcal{B}$ against that asymmetric Waters signature scheme. We note that $\mathcal{B}$ generated the parameters for the commitments for the proof Π of knowledge of M and r so that it can extract the values.

– $\mathsf{Setup}(1^k)$: we first run the $\mathsf{Setup}_{\mathcal{S}}(1^k)$ algorithm, and get $\mathsf{param}_{\mathcal{S}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, h_1, \boldsymbol{u})$. We set $\mathsf{param}_s = \mathsf{param}_{\mathcal{S}}$, and $\mathsf{param}_e = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$. $\mathcal{B}$ sets the commitment parameters so that it can extract committed values.
– $\mathsf{EKeyGen}(\mathsf{param}_e)$: for each new key request, $\mathcal{B}$ chooses one random scalar $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and defines the secret key $\mathsf{dk} = \alpha$, and the public key as $\mathsf{pk} = U_1 = g_1^\alpha$.
– $\mathsf{SKeyGen}(\mathsf{param}_s)$: for the unique signing key request, one gets the verification key $\mathsf{vk}_{\mathcal{S}}$ from the asymmetric Waters EUF-CMA security game. $\mathcal{B}$ sets $\mathsf{vk} = \mathsf{vk}_{\mathcal{S}}$.
– $\mathcal{A}$ can now access a signing oracle, with queries of the form $\mathsf{Sign}(\mathsf{vk}, \mathsf{pk}, \cdot)$, for any pk and ciphertext of its choice. But the ciphertext looks like $c = (c_1, c_2)$ together with $\Pi = (\Pi_M, \Pi_r)$.
  - If the tuple (c, Π) is not valid, then $\mathcal{B}$ returns $\perp$;
  - Otherwise, $\mathcal{B}$ can extract M from the bit-by-bit proof of knowledge Π_M (which contains the extractable commitment C_M) as well as r from Π_r (which contains the extractable commitment C_r). It then queries $\mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}_{\mathcal{S}}, M)$ to the signing oracle, and add (vk, M) to the SM set. It receives back $\sigma' = (\sigma'_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma'_2 = g_1^{-s}, \sigma'_3 = g_2^{-s})$.
    It then returns

$$\sigma = \begin{pmatrix} \sigma_1 = \sigma'_1 {\sigma'_2}^{-\alpha r} = \mathsf{sk} \cdot \mathcal{F}(M)^s \cdot U_1^{sr} = \mathsf{sk} \cdot c_1^s, \\ \sigma_2 = {\sigma'_2}^{-r} = g_1^{sr} = c_2^s, \\ \sigma_3 = {\sigma'_2}^{-\alpha} = U_1^s, \qquad \sigma_4 = {\sigma'_2}^{-1} = g_1^s, \qquad \sigma_5 = {\sigma'_3}^{-1} = g_2^s. \end{pmatrix}$$

– After a polynomial number of queries, $\mathcal{A}$ outputs, with non-negligible probability, a valid signature σ on a valid ciphertext (c, Π). As above, one can extract the message M, and for a valid forgery, one needs $M \neq \perp$ and $(M, \mathsf{vk}) \notin \mathsf{SM}$. Using SigExt, as shown above, one thus gets a valid asymmetric Waters signature on M: $\sigma_{\mathcal{B}} = (\sigma_1/\sigma_2^\alpha = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_4^{-1} = g_1^{-s}, \sigma_5^{-1} = g_2^{-s})$. This breaks the EUF-CMA property of the asymmetric Waters signature scheme, that holds under the CDH^+ assumption.

□

D Revisited Asymmetric Waters Signature on ElGamal Ciphertexts

Combining SXDH-based Groth-Sahai proofs with ElGamal encryption and asymmetric Waters signatures has the same drawback as our first instantiation from Section 3: bit-by-bit commitments of random coins.

But again, we can significantly improve efficiency of the scheme by getting rid of this extractable scalar commitment and replacing it by extractable commitments to group elements. To do so, we combine our two variants of Waters signatures: we give an *asymmetric* version of the *chosen-extended-message* secure variant from Section 4.1: in the unforgeability game the adversary has to submit extended messages of the form $(M, R_1 = g_1^r, T = X_1^r)$ to the signing oracle, which replies with the tuple $(Y \cdot (\mathcal{F}(M)R_1)^s, g_1^{-s}, g_2^{-s}, R_1^{-s})$, where $\mathsf{sk} = Y = h_1^x$ and $\mathsf{vk} = (X_1 = g_1^x, X_2 = g_2^x)$. Hence the need of X_1.

Revisited Asymmetric Waters Signature

– $\mathsf{Setup}(1^k)$: The scheme needs 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 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.
  We will sign messages $M = (M_1, \ldots, M_k) \in \{0,1\}^k$. To this aim, we need a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}_1^{k+1}$, and for convenience, we denote the *Waters Hash* as $\mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$.
  We also need an additional generator $h_1 \xleftarrow{\$} \mathbb{G}_1$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \boldsymbol{u})$.
– $\mathsf{SKeyGen}(\mathsf{param})$: Choose a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (X_1, X_2) = (g_1^x, g_2^x)$, and the secret key as $\mathsf{sk} = Y = h^x$.

– $\mathsf{Sign}(\mathsf{sk} = Y, M, R, T; s)$: First check the consistency of (R, T): $e(T, g_2) = e(R_1, X_2)$ which guarantees the expected relation on the exponents.
For some random $s \xleftarrow{\$} \mathbb{Z}_p$, define the signature as $\sigma = \big(\sigma_1 = Y(\mathcal{F}(M)R_1)^s, \sigma_2 = g_1^{-s}, \sigma_3 = g_2^{-s}, \sigma_4 = R_1^{-s}\big)$.
– $\mathsf{Verif}(\mathsf{vk} = (X_1, X_2), M, R_1, T, \sigma)$: One checks whether

$$e(\sigma_1, g_2) \cdot e(\mathcal{F}(M)R_1, \sigma_3) = e(h_1, X_2), e(\sigma_2, g_2) = e(g_1, \sigma_3) \text{ and } e(\sigma_4, g_2) = e(R_1, \sigma_3).$$

Theorem 23. *The revisited asymmetric Waters signature scheme is existentially unforgeable under chosen-extended-message attack if* CDH^+ *assumption holds.*

Proof. Let $\mathcal{A}$ be an adversary breaking the existential unforgeability of the above signature scheme, i.e. after at most q_s signing queries, it succeeds in building a new signature with probability ϵ. Let $(g_1, g_2, X_1 = g_1^a, X_2 = g_2^a, Y_1 = g_1^b)$ be an CDH^+-instance. We show how an adversary $\mathcal{B}$ can compute g_1^{ab} thanks to $\mathcal{A}$.

$\mathsf{Setup}_{\mathcal{S}}$. Pick a random position $j \xleftarrow{\$} \{0, \ldots, k\}$, choose random indices $y_0, y_1, \ldots, y_k \xleftarrow{\$} \{0, \ldots, 2\ell - 1\}$, and random scalars $z_0, z_1, \ldots, z_k \xleftarrow{\$} \mathbb{Z}_p$. One defines $X = (X_1, X_2)$, $h_1 = Y_1$, $u_0 = h_1^{y_0 - 2j\ell} g_1^{z_0}$, $u_i = h_1^{y_i} g_1^{z_i}$.

Signing queries. To answer a signing query on a message $M = (M_i)$, we define

$$H = -2j\ell + y_0 + \sum_i y_i M_i, \quad J = z_0 + \sum_i z_i M_i : \mathcal{F}(M) = h_1^H g_1^J.$$

If $H \equiv 0 \pmod p$ then abort, otherwise set $\sigma = (X_1^{-J/H} T^{-1/H} (\mathcal{F}(M)R_1)^s, X_1^{1/H} g_1^{-s}, X_2^{1/H} g_2^{-s}, T^{1/H} R_1^{-s})$. Defining $\tilde{s} = s - a/H$, we have:

$$\begin{aligned}
\sigma &:= \big(X_1^{-J/H} T^{-1/H} (h_1^H g_1^J R_1)^s,\ X_1^{1/H} g_1^{-s},\ X_2^{1/H} g_2^{-s},\ T^{1/H} R_1^{-s}\big) \\
&= \big(X_1^{-J/H} T^{-1/H} (h_1^a g_1^{Ja/H} R_1^{ta/H})(\mathcal{F}(M)R_1)^{\tilde{s}},\ X_1^{1/H} g_1^{-a/H} g_1^{-\tilde{s}},\ X_2^{1/H} g_2^{-a/H} g_2^{-\tilde{s}},\ T^{1/H} R_1^{-s}\big) \\
&= \big(h_1^a (\mathcal{F}(M)R_1)^{\tilde{s}},\ g_1^{-\tilde{s}},\ g_2^{-\tilde{s}},\ R_1^{-\tilde{s}}\big).
\end{aligned}$$

After at most q_s signing queries $\mathcal{A}$ outputs a forgery $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*))$ on M^*. As before, we define

$$H^* = -2j\ell + y_0 + \sum_i y_i M_i^*, \quad J^* = z_0 + \sum_i z_i M_i^* : \mathcal{F}(M^*) = h_1^{H^*} g_1^{J^*}.$$

If $H^* \not\equiv 0 \pmod p$ then abort, otherwise, for some s^*, $\sigma^* = (h_1^a (\mathcal{F}(M^*)R_1)^{s^*}, g_1^{-s^*}, g_2^{-s^*}, R_1^{-s^*})$, and thus $\sigma^* = (h_1^a g_1^{J^* s^*} R_1^{s^*}, g_1^{-s^*}, g_2^{-s^*}, R_1^{-s^*})$. As a consequence, $\sigma_1^* (\sigma_2^*)^{J^*} / \sigma_4 = h_1^a = g_1^{ab}$: one has solved the CDH^+ problem. As above, the latter case occurs with good probability. □

Once again, we can remark that signing and verifying can be performed without knowing the message M itself, but $F = \mathcal{F}(M)$ only is enough. We can thus replace M by (F, Π_M).

Theorem 24. *The revisited asymmetric Waters signature scheme is randomizable when one defines* $\mathsf{Random}(\mathsf{vk}, (F, \Pi_M), R_1, T, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4); s')$ *to output* $\sigma' = (\sigma_1 \cdot (FR_1)^{s'}, \sigma_2 \cdot g_1^{-s'}, \sigma_3 \cdot g_2^{-s'}, \sigma_4 \cdot R_1^{-s'})$*, for random* $s' \xleftarrow{\$} \mathbb{Z}_p$.

Proof. This is indeed valid re-randomization:

$$\sigma = \mathsf{Sign}(\mathsf{sk}, (F, \Pi_M), R, T; s) \Rightarrow \mathsf{Random}(\mathsf{vk}, (F, \Pi_M), R, T, \sigma; s') = \mathsf{FSign}(\mathsf{sk}, F, R, T, \Pi_M; s + s').$$

□

Signature on Encrypted Messages We now want to generate a signature on an encryption of $\mathcal{F}(M)$. Our improved construction just differs for the proof Π_r: We now have $C_r = \mathcal{C}(X_1^r)$ which is here to help the simulator in creating T in the reduction, together with proof Π_r showing that $e(\langle C\rangle, g_2) = e(c_2, X_2)$ and so that C is a commitment of X_1 raised to r: This equation is a linear pairing product equation. Therefore it requires 2 in $\mathbb{G}_2$.

We thus need 1 commitment, so 2 group elements in $\mathbb{G}_1$, and the proof which requires 2 in $\mathbb{G}_2$ instead of 6ℓ group elements in $\mathbb{G}_1$ and $6\ell + 2$ in $\mathbb{G}_2$.

Π_M is constructed exactly as previously, with the same verification equations. therefore Π will be composed of $6k + 2$ elements in $\mathbb{G}_1$ and $6k + 4$ group elements in $\mathbb{G}_2$.

Theorem 25. *Our variant of Asymmetric Waters signature on ElGamal ciphertexts is randomizable and unforgeable (in the UF sense) under the* CDH^+ *assumption.*

Proof. Let us denote $\mathcal{SC}$ our above signature on ciphertexts (but omit it in the subscripts for clarity), and $\mathcal{S}$ the asymmetric Waters signature scheme. We know that the latter is existentially unforgeable under the CDH^+ assumption. Let us assume that $\mathcal{A}$ is able to break the unforgeabilty of $\mathcal{SC}$. We will build an adversary $\mathcal{B}$ against that asymmetric Waters signature scheme. We note that $\mathcal{B}$ generated the parameters for the commitments for the proof Π of knowledge of M, r, so that it can extract the values.

- $\mathsf{Setup}(1^k)$: we first run the $\mathsf{Setup}_{\mathcal{S}}(1^k)$ algorithm, and get $\mathsf{param}_{\mathcal{S}} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2, h_1, \boldsymbol{u})$. We set $\mathsf{param}_s = \mathsf{param}_{\mathcal{S}}$, and $\mathsf{param}_e = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e, g_1, g_2)$. $\mathcal{B}$ sets the commitment parameters so that it can extract committed values.
- $\mathsf{EKeyGen}(\mathsf{param}_e)$: for each new key request, $\mathcal{B}$ chooses one random scalar $\alpha \xleftarrow{\$} \mathbb{Z}_p$, and defines the secret key $\mathsf{dk} = \alpha$, and the public key as $\mathsf{pk} = U_1 = g_1^\alpha$.
- $\mathsf{SKeyGen}(\mathsf{param}_s)$: for the unique signing key request, one gets the verification key $\mathsf{vk}_{\mathcal{S}}$ from the Waters EUF-CMA security game. $\mathcal{B}$ sets $\mathsf{vk} = \mathsf{vk}_{\mathcal{S}}$.
- $\mathcal{A}$ can now access a signing oracle, with queries of the form $\mathsf{Sign}(\mathsf{vk}, \mathsf{pk}, \cdot)$, for any pk and ciphertext of its choice. But the ciphertext looks like $c = (c_1, c_2)$ together with $\Pi = (\Pi_M, \Pi_r)$.
  - If the tuple (c, Π) is not valid, then $\mathcal{B}$ returns $\perp$;
  - Otherwise, $\mathcal{B}$ can extract M from the proof of knowledge Π_M that contains the extractable bit-by-bit commitment C_M. $\mathcal{B}$ can also compute $R_1 = g_1^r$, and extract $T = X_1^r$ from Π_r, that contains the extractable (not bit-by-bit) commitment $C_r = \mathcal{C}(X_1^r)$. It then queries $\mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}_{\mathcal{S}}, M, R_1, T)$ to the signing oracle, and adds (vk, M) to the SM set. It receives back $\sigma' = (\sigma'_1 = \mathsf{sk}(\mathcal{F}(M)R_1)^s, \sigma'_2 = g_1^{-s}, \sigma'_3 = g_2^{-s}, \sigma'_4 = R_1^{-s})$. It then returns

$$\sigma = \begin{pmatrix} \sigma_1 = \sigma'_1 \cdot {\sigma'_4}^{1-\alpha} = \mathsf{sk} \cdot \mathcal{F}(M)^s \cdot U_1^{sr} = \mathsf{sk} \cdot c_1^s, & \\ \sigma_2 = {\sigma'_4}^{-1} = g_1^{sr} = c_2^s, & \sigma_3 = {\sigma'_2}^{-\alpha} = U_1^s, \\ \sigma_4 = {\sigma'_2}^{-1} = g_1^s, & \sigma_5 = {\sigma'_3}^{-1} = g_2^s \end{pmatrix}.$$

- After a polynomial number of queries, $\mathcal{A}$ outputs, with non-negligible probability, a valid signature σ on a valid ciphertext (c, Π). As above, one can extract the message M, and for a valid forgery, one needs $M \neq \perp$ and $(M, \mathsf{vk}) \notin \mathsf{SM}$. Using SigExt, as shown above, one thus gets a valid asymmetric Waters signature on M: $\sigma_{\mathcal{B}} = (\sigma_1/\sigma_2^\alpha = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_4^{-1} = g_1^{-s}, \sigma_5^{-1} = g_2^{-s})$. This breaks the EUF-CMA property of our variant of the asymmetric Waters signature scheme, that holds under the CDH^+ assumption.

□