# 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

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

D. Catalano et al. (Eds.): PKC 2011, LNCS 6571, pp. 403–422, 2011.

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, 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 available in the full version [BFPV11]. 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 $\mathsf{CDH}^+$, 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 | $9k + 33$ |

| Asymmetric Pairing | $\mathbb{G}_1$ | $\mathbb{G}_2$ |
|---|---|---|
| Waters + ElGamal | $6k + 15$ | $6k + 7$ |

*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 full version [BFPV11]. 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* $\mathsf{param}$ *of the scheme;*
- $\mathsf{EKeyGen}(\mathsf{param})$ *generates a pair of keys, the public (encryption) key* $\mathsf{pk}$ *and the associated private (decryption) key* $\mathsf{dk}$*;*
- $\mathsf{Encrypt}(\mathsf{pk}, m; r)$ *produces a ciphertext* $c$ *on the input message* $m \in \mathcal{M}$ *and the public key* $\mathsf{pk}$*, using the random coins* $r \in \mathcal{R}_e$*;*
- $\mathsf{Decrypt}(\mathsf{dk}, c)$ *decrypts the ciphertext* $c$ *under the private key* $\mathsf{dk}$*; 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* $\mathsf{param}$ *of the scheme;*
- $\mathsf{SKeyGen}(\mathsf{param})$ *generates a pair of keys, the public (verification) key* $\mathsf{vk}$ *and the private (signing) key* $\mathsf{sk}$*;*
- $\mathsf{Sign}(\mathsf{sk}, m; s)$ *produces a signature* $\sigma$ *on the input message* $m$*, under the signing key* $\mathsf{sk}$*, and using the random coins* $s \in \mathcal{R}_s$*;*
- $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$ *checks whether* $\sigma$ *is a valid signature on* $m$*, w.r.t. the public key* $\mathsf{vk}$*; 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 $(F, \Pi_M)$, where $\Pi_M$ 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* $\mathsf{param}_e$ *and* $\mathsf{param}_s$ *for the associated encryption and signature schemes;*
- $\mathsf{EKeyGen}(\mathsf{param}_e)$ *generates a pair of keys, the encryption key* $\mathsf{pk}$ *and the associated decryption key* $\mathsf{dk}$*;*
- $\mathsf{SKeyGen}(\mathsf{param}_s)$ *generates a pair of keys, the verification key* $\mathsf{vk}$ *and the signing key* $\mathsf{sk}$*;*
- $\mathsf{Encrypt}(\mathsf{pk}, \mathsf{vk}, m, r)$ *produces a ciphertext* $c$ *on input the message* $m \in \mathcal{M}$ *and the encryption key* $\mathsf{pk}$*, 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* $\mathsf{vk}$ *(the field for* $\mathsf{vk}$ *can be empty if the signing algorithm is universal and does not require a ciphertext specific to the signer);*

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

- $\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, c; s)$*, on input a ciphertext* $c$ *and a signing key* $\mathsf{sk}$*, using the random coins* $s \in \mathcal{R}_s$*, produces a signature* $\sigma$*, or* $\perp$ *if the ciphertext* $c$ *is not valid (w.r.t.* $\mathsf{pk}$*, and possibly* $\mathsf{vk}$ *associated to* $\mathsf{sk}$*);*
- $\mathsf{Decrypt}(\mathsf{dk}, \mathsf{vk}, c)$ *decrypts the ciphertext* $c$ *under the private key* $\mathsf{dk}$*. It outputs the plaintext, or* $\perp$ *if* $c$ *is invalid (w.r.t.* $\mathsf{pk}$*, and possibly* $\mathsf{vk}$*);*
- $\mathsf{Verif}(\mathsf{vk}, \mathsf{pk}, c, \sigma)$ *checks whether* $\sigma$ *is a valid signature on* $c$*, w.r.t. the public key* $\mathsf{vk}$*. It outputs 1 if the signature is valid, and 0 otherwise (possibly because of an invalid ciphertext* $c$*, with respect to* $\mathsf{pk}$*, and possibly* $\mathsf{vk}$*).*

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 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 $\mathsf{pk}_i$ and a ciphertext $c$, and generates a signature $\sigma$ on it (if the ciphertext is valid). It also updates the set $\mathsf{SM}$ 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* $\mathsf{pk}$*, 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* $\sigma'$ *valid under* $\mathsf{vk}$ *from* $\sigma$ *on a message* $M$ *given as* $F = \mathcal{F}(M)$ *and a proof* $\Pi_M$ *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* $\Pi_M$ *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* $\mathsf{pk}$*, and a signature* $\sigma'$ *on* $c'$*. Further inputs are a signature* $\sigma$ *on* $c$ *under* $\mathsf{vk}$*, 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 $\sigma'$.

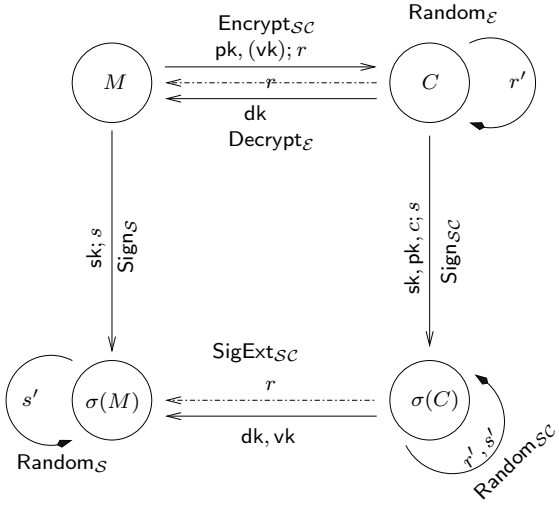
Let us assume that there is a signature scheme $\mathcal{S}$ where $\mathsf{Setup}_{\mathcal{S}}$ is the projection of $\mathsf{Setup}_{\mathcal{SC}}$ on the signature component and $\mathsf{SKeyGen}_{\mathcal{S}} = \mathsf{SKeyGen}$. The scheme $\mathcal{SC}$ is *extractable* if the following holds: 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 $\mathsf{vk}$, 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 $\sigma$ on the ciphertext $c$. From $(c, \sigma)$ the owner of the decryption key can now not only recover the encrypted message $m$, but also a signature $\sigma'$ on the *message* $m$, using the functionality $\mathsf{SigExt}_{\mathcal{SC}}$. The signature $\sigma$ 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 coins $r$ used for the encryption. In all our instantiations we have that $\sigma$ is the encryption of $\sigma'$ with the same coins $r$ used to encrypt the message. The user who encrypted $m$ is thus able to extract $\sigma'$, 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).




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

## 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 the full version [BFPV11].

### 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 the full version [BFPV11]. 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 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 $(M, \sigma)$. As a consequence, if we work directly with $\mathcal{F}(M)$, we will need to add a proof of knowledge $\Pi_M$ 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.** The secret key $\mathsf{dk}$ 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 $c = \left(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot M\right)$, 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 $\mathsf{DLin}$ 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 $\Pi_M$ 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 $c = \left(c_1 = X_1^{r_1}, c_2 = X_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \mathcal{F}(M)\right)$ 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 $\Pi_M$ 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: $\Pi_r$ 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 $\ell$ 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 $\ell$ are the respective bit lengths of messages and of the

order of $\mathbb{G}$). Such an extended ciphertext $(c, \Pi)$ can then be signed, after a test of validity of the proof $\Pi$. 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 $\mathsf{CDH}$ 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 $\Pi_M$ that contains commitments to the bits of message $M$, the knowledge of the trapdoor $(\lambda, \mu)$ 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 the full version [BFPV11].

## 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 $M$, $\Pi_M$ (composed of $C_M$ and a proof) consists of $9k + 2$ group elements. The random coins $r_1$ and $r_2$ being $\ell$-bit long, $\Pi_r$ (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" $(M, R_1 := g^{r_1}, R_2 := g^{r_2}, T := \mathsf{vk}^{r_1+r_2})$ 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 $\mathsf{sk} = Z = h^y$.
- $\mathsf{Sign}(\mathsf{sk} = Z, M, R_1, R_2, T; s)$: First check the consistency of $(R_1, R_2, T)$: if $e(R_1 R_2, Y) = e(g, T)$ then this guarantees that there exists $(r_1, r_2)$ such that $R_1 = g^{r_1}, R_2 = g^{r_2}$, $T = Y^{r_1 + r_2}$. Choose a random $s \xleftarrow{\$} \mathbb{Z}_p$ and define the signature as $\sigma = \big(\sigma_1 = Z \cdot (\mathcal{F}(M) R_1 R_2)^s, \sigma_2 = g^{-s}, \sigma_3 = R_1^{-s}, \sigma_4 = R_2^{-s}\big)$. Again, we may replace the input message $M$ by the pair $(\mathcal{F}(M), \Pi_M)$.
- $\mathsf{Verif}(\mathsf{vk} = Y, M, R_1, R_2, T, \sigma)$: Check whether $e(g, \sigma_1) \cdot e(\mathcal{F}(M) R_1 R_2, \sigma_2) = e(Y, h)$, $e(g, \sigma_3) = e(\sigma_2, R_1)$ and $e(g, \sigma_4) = e(\sigma_2, R_2)$, as well as the consistency of $(R_1, R_2, T)$, via $e(R_1 R_2, Y) = e(g, T)$.

To randomize a signature, we define $\mathsf{Random}(\mathsf{vk}, (F, \Pi_M), R_1, R_2, T, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4); s')$ to output $\sigma' = (\sigma_1 \cdot (F R_1 R_2)^{s'}, \sigma_2 \cdot g^{-s'}, \sigma_3 \cdot R_1^{-s'}, \sigma_4 \cdot R_2^{-s'})$, 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 $\sigma'$ 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 the full version [BFPV11].

### 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 $\mathsf{param}_e$ 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 $\Pi_r$ contains the commitments $C_r = (C_1 = \mathcal{C}'(r_1), C_2 = \mathcal{C}'(r_2), C_3 = \mathcal{C}(Y^{r_1+r_2}))$, from which the simulator can extract $R_1, R_2$ and $T$ in the reduction (see below). $\Pi_r$ moreover contains proofs of consistency: $C_3$ is a commitment to $Y$ raised to the values $r_1$ and $r_2$ committed in $C_1$ and $C_2$, which in turn are the coins for the encryption: $c_1 = X_1^{\langle r_1 \rangle}$, $c_2 = X_2^{\langle r_2 \rangle}$, and $\langle Y^{r_1+r_2} \rangle = Y^{\langle r_1 \rangle + \langle r_2 \rangle}$. These equations are multi-scalar multiplication equations, from which only the last one is non-linear. We require 9 group elements for the commitments and 13 for the proofs, thus 22 group elements instead of $18\ell + 4$ in the previous construction.
  - Proof $\Pi_M$ 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: $c_3 = (u_0 \prod_{i \in \{1,\ldots,k\}} u_i^{\langle M_i \rangle}) \cdot g^{\langle r_1 \rangle + \langle r_2 \rangle}$, which is a linear multi-scalar multiplication equation (2 additional group elements). $\Pi_M$ is therefore composed of $9k + 2$ group elements.

  The global proof (containing the commitments) $\Pi$ consists therefore of $9k + 24$ group elements (instead of $9k + 18\ell + 6$ when using the original Waters scheme), where $k$ and $\ell$ 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 $\Pi$ 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 $\Pi$), 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 $\Pi$ and checks whether the following pairing equations hold: $e(\sigma_3, g) = e(h, Y) \cdot e(c_3, \sigma_6)$ and

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

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

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

together with a randomization $\Pi'$ of $\Pi$.

– $\mathsf{SigExt}\big(\mathsf{dk} = (x_1, x_2), \mathsf{vk}, (c = (c_1, c_2, c_3), \Pi), \sigma\big)$: 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 $M$:

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

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 $\mathsf{Random}$ follows from inspection of the construction of $c'$ and $\sigma'$ 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 the full version [BFPV11]. As motivated when introducing the additional elements in the signature query, from a valid signing query, our simulator can extract $M$, but also $R_1 = g^{r_1}$, $R_2 = g^{r_2}$ and $T = Y^{r_1+r_2}$ from the commitments (but not the scalars $r_1$ and $r_2$). It adds $M$ to the set $\mathsf{SM}$ and then queries $\mathsf{Sign}_{\mathcal{S}}(\mathsf{sk}, M, R_1, R_2, T)$ 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:

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

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 $\sigma$ on $c$. The voting center can now consistently randomize both $c$ and $\sigma$ as $c'$ and $\sigma'$, 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 42 group elements in the instantiation with linear encryption, and even smaller for the instantiation using ElGamal detailed in the full version [BFPV11]: 21 $\mathbb{G}_1$ elements and 13 $\mathbb{G}_2$ 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 $\sigma$, 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 $\sigma$ to the transcript. However, due to the randomizability of the signature, the user can randomize $\sigma$ into $\sigma'$ that is a secure blind signature:

- the blindness property relies on the semantic security of the encryption scheme (here $\mathsf{DLin}$) and the randomization, which is information-theoretic;
- the one-more unforgeability relies on unforgeability of the signature scheme (here $\mathsf{CDH}$), 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] Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure-preserving signatures and commitments to group elements. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 209–236. Springer, Heidelberg (2010)

[BBS04] Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)

[BCC+09] Belenkiy, M., Camenisch, J., Chase, M., Kohlweiss, M., Lysyanskaya, A., Shacham, H.: Randomizable proofs and delegatable anonymous credentials. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 108–125. Springer, Heidelberg (2009)

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

[BFPV11] Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Gennaro, R. (ed.) Proceedings of PKC 2011. LNCS, vol. 6571. Springer, Heidelberg (2010), Full version available from the web page of the authors

[BNPS01] Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The Power of RSA Inversion Oracles and the Security of Chaum's RSA-Based Blind Signature Scheme. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 309–338. Springer, Heidelberg (2002)

[BR93] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Ashby, V. (ed.) ACM CCS 1993: 1st Conference on Computer and Communications Security, pp. 62–73. ACM Press, New York (1993)

[BW06] Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)

[Cha83] Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology – CRYPTO 1982, pp. 199–203. Plenum Press, New York (1983)

[DFN06] Damgård, I., Fazio, N., Nicolosi, A.: Non-interactive Zero-Knowledge from Homomorphic Encryption. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 41–59. Springer, Heidelberg (2006)

[Fis06] Fischlin, M.: Round-optimal composable blind signatures in the common reference string model. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 60–77. Springer, Heidelberg (2006)

[FP09] Fuchsbauer, G., Pointcheval, D.: Proofs on Encrypted Values in Bilinear Groups and an Application to Anonymity of Signatures. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 132–149. Springer, Heidelberg (2009)

[Fuc09] Fuchsbauer, G.: 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] Fuchsbauer, G.: 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] Gjøsteen, K., Kråkmo, L.: Round-Optimal Blind Signatures from Waters Signatures. In: Baek, J., Bao, F., Chen, K., Lai, X. (eds.) ProvSec 2008. LNCS, vol. 5324, pp. 112–126. Springer, Heidelberg (2008)

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

[GS08] Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (2008)

[MSF10] Meiklejohn, S., Shacham, H., Freeman, D.M.: Limitations on Transformations from Composite-Order to Prime-Order Groups: The Case of Round-Optimal Blind Signatures. In: Abe, M. (ed.) ASIACRYPT 2010. LNCS, vol. 6477, pp. 519–538. Springer, Heidelberg (2010)

[PS96] Pointcheval, D., Stern, J.: Provably secure blind signature schemes. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 252–265. Springer, Heidelberg (1996)

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

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

[SPC95] Stadler, M.A., Piveteau, J.-M., Camenisch, J.: Fair Blind Signatures. In: Guillou, L.C., Quisquater, J.-J. (eds.) EUROCRYPT 1995. LNCS, vol. 921, pp. 209–219. Springer, Heidelberg (1995)

[Wat05] Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)