Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions*

Olivier Blazy, David Pointcheval, and Damien Vergnaud

ENS, Paris, France

Abstract. In 2008, Groth and Sahai proposed a powerful suite of techniques for constructing non-interactive zero-knowledge proofs in bilinear groups. Their proof systems have found numerous applications, including group signature schemes, anonymous voting, and anonymous credentials. In this paper, we demonstrate that the notion of *smooth projective hash functions* can be useful to design round-optimal privacy-preserving interactive protocols. We show that this approach is suitable for designing schemes that rely on standard security assumptions in the standard model with a common-reference string and are more efficient than those obtained using the Groth-Sahai methodology. As an illustration of our design principle, we construct an efficient oblivious signature-based envelope scheme and a blind signature scheme, both round-optimal.

1 Introduction

In 2008, Groth and Sahai [22] proposed a way to produce efficient and practical non-interactive zero-knowledge and non-interactive witness-indistinguishable proofs for (algebraic) statements related to groups equipped with a bilinear map. They have been significantly studied in cryptography and used in a wide variety of applications in recent years (*e.g.* group signature schemes [8, 9, 20] or blind signatures [2, 5]). While avoiding expensive NP-reductions, these proof systems still lack in practicality and it is desirable to provide more efficient tools.

Smooth projective hash functions (SPHF) were introduced by Cramer and Shoup [13] for constructing encryption schemes. A projective hashing family is a family of hash functions that can be evaluated in two ways: using the (secret) hashing key, one can compute the function on every point in its domain, whereas using the (public) *projected* key one can only compute the function on a special subset of its domain. Such a family is deemed *smooth* if the value of the hash function on any point outside the special subset is independent of the projected key. If it is hard to distinguish elements of the special subset from non-elements, then this primitive can be seen as special type of zero-knowledge proof system for membership in the special subset. The notion of SPHF has found applications in various contexts in cryptography (*e.g.* [18, 26, 1]). We present some other applications with privacy-preserving primitives that were already inherently interactive.

* CNRS – UMR 8548 and INRIA – EPI Cascade, Université Paris Diderot.

Applications: Our two applications are *Oblivious Signature-Based Envelope* [27] and *Blind Signatures* [12].

Oblivious Signature-Based Envelope (OSBE) were introduced in [27]. It can be viewed as a nice way to ease the asymmetrical aspect of several authentication protocols. Alice is a member of an organization and possesses a certificate produced by an authority attesting she is in this organization. Bob wants to send a private message P to members of this organization. However due to the sensitive nature of the organization, Alice does not want to give Bob neither her certificate nor a proof she belongs to the organization. OSBE lets Bob sends an obfuscated version of this message P to Alice, in such a way that Alice will be able to find P if and only if Alice is in the required organization. In the process, Bob cannot decide whether Alice does really belong to the organization. They are part of a growing field of protocols, around *automated trust negotiation*, which also include Secret Handshakes [3], Password-based Authenticated Key-Exchange [19], and Hidden Credentials [10]. Those schemes are all closely related, so due to space constraints, we are going to focus on OSBE (as if you tweak two of them, you can produce any of the other protocols [11]).

Blind signatures were introduced by Chaum [12] for electronic cash in order to prevent the bank from linking a coin to its spender: they allow a user to obtain a signature on a message such that the signer cannot relate the resulting message/signature pair to the execution of the signing protocol. In [15], Fischlin gave a generic construction of round-optimal blind signatures in the common-reference string (CRS) model: the signing protocol consists of one message from the user to the signer and one response by the signer. The first practical instantiation of round-optimal blind signatures in the standard model was proposed in [2] but it relies on non-standard computational assumptions. We proposed, recently only [5], the most efficient realizations of round-optimal blind signatures in the common-reference string model under classical assumptions. But these schemes still use the Groth-Sahai proof systems.

Contributions: Our first contribution is to clarify and increase the security requirements of an OSBE scheme. The main improvement residing in some protection for both the sender and the receiver against the Certification Authority. The OSBE notion echoes directly to the idea of SPHF if we consider the language $\mathcal{L}$ defined by encryption of valid signatures, which is hard to distinguish under the security of the encryption schemes. We show how to build, from a SPHF on this language, an OSBE scheme in the standard model with a CRS. And we prove the security of our construction in regards of the security of the commitment (the ciphertext), the signature and the SPHF scheme. We then show how to build a simple and efficient OSBE scheme relying on a classical assumption, DLin. An asymmetrical version is available in the full version [6]. To build those schemes, we use SPHF in a new way, avoiding the need of costly Groth-Sahai proofs when an interaction is inherently needed in the primitive. Our method does not add any other interaction, and so supplement smoothly those proofs.

To show the efficiency of the method, and the ease of application, we then adapt two Blind Signature schemes proposed in [5]. Our approach fits perfectly and decreases significantly the communicational complexity of the schemes (it is divided by more than three in one construction). Moreover one scheme relies on a weakened security assumptions: the XDH assumption instead of the SXDH assumption and permits to use more bilinear group settings (namely, Type-II and Type-III bilinear groups [16] instead of only Type-III bilinear groups for the construction presented in [5]).

2 Definitions

In this section, we briefly recall the notations and the security notions of the basic primitives we will use in the rest of the paper, and namely public key encryption, signature and smooth projective hash functions (SPHF), using the Gennaro-Lindell [18] extension. More details are available in the full version [6]. In a second part, we recall and enhance the security model of oblivious signature-based envelope protocols [27].

2.1 Notations

Encryption Scheme. A (public-key) encryption scheme is defined by four algorithms: $\mathsf{param} \leftarrow \mathsf{ESetup}(1^k)$, $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{EKeyGen}(\mathsf{param})$, $c \leftarrow \mathsf{Encrypt}(\mathsf{ek}, m; r)$, and $m \leftarrow \mathsf{Decrypt}(\mathsf{dk}, c)$. We will need the classical notion of IND-CPA security. More precisely, we will use commitment schemes (as in [1]), which should be hiding (indistinguishability) and binding (one opening only), with the additional extractability property. The latter property thus needs an extracting algorithm that corresponds to the decryption algorithm. Hence the notation with encryption schemes.

Signature Scheme. A signature scheme is defined by four algorithms: $\mathsf{param} \leftarrow \mathsf{SSetup}(1^k)$, $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m; s)$, and $\mathsf{Verif}(\mathsf{vk}, m, \sigma)$. We will need the classical notion of EUF-CMA security.

Smooth Projective Hash Function. An SPHF system [13] on a language $\mathcal{L}$ is defined by five algorithms: $\mathsf{SPHFSetup}(1^k)$ that generates the global parameters, $\mathsf{HashKG}(\mathcal{L}, \mathsf{param})$ that generates a hashing key hk, $\mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$ that derives the projection key hp, possibly depending on the word W [18, 1]. Then, $\mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), W)$ and $\mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), W, w)$ outputs the hash value, either from the hashing key, or from the projection key and the witness. The correctness of the scheme assures that if W is indeed in $\mathcal{L}$ with w as a witness, then the two ways to compute the hash value give the same result. The security of a SPHF is defined through two different notions, the smoothness and the pseudo-randomness properties: The smoothness property guarantees that if $W \notin \mathcal{L}$, then the hash value is statistically random (statistically indistinguishable from a random element). The pseudo-randomness guarantees that even for a word $W \in \mathcal{L}$, but without the knowledge of a witness w, then the hash value is random (computationally indistinguishable from a random element). Abdalla *et al.* [1] explained how to combine SPHF to deal with conjunctions and disjunctions of the languages.

2.2 Oblivious Signature-Based Envelope

We now define an OSBE protocol, where a sender $\mathcal{S}$ wants to send a private message $P \in \{0,1\}^\ell$ to a recipient $\mathcal{R}$ in possession of a certificate/signature on a message M.

Definition 1 (Oblivious Signature-Based Envelope). *An OSBE scheme is defined by four algorithms* $(\mathsf{OSBESetup}, \mathsf{OSBEKeyGen}, \mathsf{OSBESign}, \mathsf{OSBEVerif})$, *and one interactive protocol* $\mathsf{OSBEProtocol}\langle\mathcal{S},\mathcal{R}\rangle$*:*

- $\mathsf{OSBESetup}(1^k)$, *where k is the security parameter, generates the global parameters* param*;*
- $\mathsf{OSBEKeyGen}(\mathsf{param})$ *generates the keys* $(\mathsf{vk}, \mathsf{sk})$ *of the certification authority;*
- $\mathsf{OSBESign}(\mathsf{sk}, m)$ *produces a signature σ on the input message m, under the signing key* sk*;*
- $\mathsf{OSBEVerif}(\mathsf{vk}, m, \sigma)$ *checks whether σ is a valid signature on m, w.r.t. the public key* vk*; it outputs 1 if the signature is valid, and 0 otherwise.*
- $\mathsf{OSBEProtocol}\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ *between the sender $\mathcal{S}$ with the private message P, and the recipient $\mathcal{R}$ with a certificate σ. If σ is a valid signature under* vk *on the common message M, then $\mathcal{R}$ receives P, otherwise it receives nothing. In any case, $\mathcal{S}$ does not learn anything.*

Such an OSBE scheme should be (the three last properties are additional —or stronger— security properties from the original definitions [27]):

- *correct*: the protocol actually allows $\mathcal{R}$ to learn P, whenever σ is a valid signature on M under vk;
- *oblivious*: the sender should not be able to distinguish whether $\mathcal{R}$ uses a valid signature σ on M under vk as input. More precisely, if $\mathcal{R}_0$ knows and uses a valid signature σ and $\mathcal{R}_1$ does not use such a valid signature, the sender cannot distinguish an interaction with $\mathcal{R}_0$ from an interaction with $\mathcal{R}_1$;
- *(weakly) semantically secure*: the recipient learns nothing about $\mathcal{S}$ input P if it does not use a valid signature σ on M under vk as input. More precisely, if $\mathcal{S}_0$ owns P_0 and $\mathcal{S}_1$ owns P_1, the recipient that does not use a valid signature cannot distinguish an interaction with $\mathcal{S}_0$ from an interaction with $\mathcal{S}_1$;
- *semantically secure* (denoted sem): the above indistinguishability should hold even if the receiver has seen several interactions $\langle\mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ with valid signatures, and the same sender's input P;
- *escrow free* (denoted esc): the authority (owner of the signing key sk), playing as the sender or just eavesdropping, is unable to distinguish whether $\mathcal{R}$ used a valid signature σ on M under vk as input. This notion supersedes the above *oblivious* property, since this is basically oblivious w.r.t. the authority, without any restriction.
- *semantically secure w.r.t. the authority* (denoted sem^*): after the interaction, the authority (owner of the signing key sk) learns nothing about P.

We insist that the escrow-free property (esc) is stronger than the oblivious property, hence we will consider the former only. However, the semantic security

```
$\mathsf{Exp}^{\mathsf{esc}-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Escrow Free property]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $\mathsf{vk} \leftarrow \mathcal{A}(\mathtt{INIT} : \mathsf{param})$
3. $(M, \sigma) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathtt{Send}(\mathsf{vk}, \cdot, \cdot), \mathtt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \mathtt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
4. $\mathsf{OSBEProtocol}\langle \mathcal{A}, \mathtt{Rec}^*(\mathsf{vk}, M, \sigma, b)\rangle$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathtt{Send}(\mathsf{vk}, \cdot, \cdot), \mathtt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \mathtt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
6. RETURN $b'$
```

```
$\mathsf{Exp}^{\mathsf{sem}^*-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Semantic security w.r.t. the authority]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $\mathsf{vk} \leftarrow \mathcal{A}(\mathtt{INIT} : \mathsf{param})$
3. $(M, \sigma, P_0, P_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathtt{Send}(\mathsf{vk}, \cdot, \cdot), \mathtt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \mathtt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
4. $\mathsf{transcript} \leftarrow \mathsf{OSBEProtocol}\langle \mathtt{Send}(\mathsf{vk}, M, P_b), \mathtt{Rec}^*(\mathsf{vk}, M, \sigma, 0\rangle$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathsf{transcript}, \mathtt{Send}(\mathsf{vk}, \cdot, \cdot), \mathtt{Rec}^*(\mathsf{vk}, \cdot, \cdot, 0), \mathtt{Exec}^*(\mathsf{vk}, \cdot, \cdot, \cdot))$
6. RETURN $b'$
```

```
$\mathsf{Exp}^{\mathsf{sem}-b}_{\mathcal{OSBE},\mathcal{A}}(k)$ [Semantic Security]
1. $\mathsf{param} \leftarrow \mathsf{OSBESetup}(1^k)$
2. $(\mathsf{vk}, \mathsf{sk}) \leftarrow \mathsf{OSBEKeyGen}(\mathsf{param})$
3. $(M, P_0, P_1) \leftarrow \mathcal{A}(\mathtt{FIND} : \mathsf{vk}, \mathtt{Sign}^*(\mathsf{vk}, \cdot), \mathtt{Send}(\mathsf{vk}, \cdot, \cdot), \mathtt{Rec}(\mathsf{vk}, \cdot, 0), \mathtt{Exec}(\mathsf{vk}, \cdot, \cdot))$
4. $\mathsf{OSBEProtocol}\langle \mathtt{Send}(\mathsf{vk}, M, P_b), \mathcal{A}\rangle$
5. $b' \leftarrow \mathcal{A}(\mathtt{GUESS} : \mathtt{Sign}(\mathsf{vk}, \cdot), \mathtt{Send}(\mathsf{vk}, \cdot, \cdot), \mathtt{Rec}(\mathsf{vk}, \cdot, 0), \mathtt{Exec}(\mathsf{vk}, \cdot, \cdot))$
6. IF $M \in \mathcal{SM}$ RETURN 0 ELSE RETURN $b'$
```

Fig. 1. Security Games for $\mathcal{OSBE}$

w.r.t. the authority (sem^*) is independent from the basic semantic security (sem) since in the latter the adversary interacts with the sender whereas in the former the adversary (who generated the signing keys) has only passive access to a challenge transcript.

These security notions can be formalized by the security games presented on Figure 1, where the adversary keeps some internal state between the various calls INIT, FIND and GUESS. They make use of the oracles described below, and the advantages of the adversary are, for all the security notions,

$$\mathsf{Adv}^*_{\mathcal{OSBE},\mathcal{A}}(k) = \Pr[\mathsf{Exp}^{*-1}_{\mathcal{OSBE},\mathcal{A}}(k) = 1] - \Pr[\mathsf{Exp}^{*-0}_{\mathcal{OSBE},\mathcal{A}}(k) = 1]$$

$$\mathsf{Adv}^*_{\mathcal{OSBE}}(k, t) = \max_{\mathcal{A} \leq t} \mathsf{Adv}^*_{\mathcal{OSBE},\mathcal{A}}(k).$$

- $\mathtt{Sign}(\mathsf{vk}, m)$: This oracle outputs a valid signature on m under the signing key sk associated to vk (where the pair $(\mathsf{vk}, \mathsf{sk})$ has been outputted by the $\mathsf{OSBEKeyGen}$ algorithm);
- $\mathtt{Sign}^*(\mathsf{vk}, m)$: This oracle first queries $\mathtt{Sign}(\mathsf{vk}, m)$. It additionally stores the query m to the list $\mathcal{SM}$;
- $\mathtt{Send}(\mathsf{vk}, m, P)$: This oracle emulates the sender with private input P, and thus may consist of multiple interactions;
- $\mathtt{Rec}(\mathsf{vk}, m, b)$: This oracle emulates the recipient either with a valid signature σ on m under the verification key vk (obtained from the signing oracle $\mathtt{Sign}$)

if $b = 0$ (as the above $\mathcal{R}_0$), or with a random string if $b = 1$ (as the above $\mathcal{R}_1$). This oracle is available when the signing key has been generated by $\mathsf{OSBEKeyGen}$ only;

– $\mathtt{Rec}^*(\mathsf{vk}, m, \sigma, b)$: This oracle does as above, with a valid signature σ provided by the adversary. If $b = 0$, it emulates the recipient playing with σ; if $b = 1$, it emulates the recipient playing with a random string;

– $\mathtt{Exec}(\mathsf{vk}, m, P)$: This oracle outputs the transcript of an honest execution between a sender with private input P and the recipient with a valid signature σ on m under the verification key vk (obtained from the signing oracle $\mathtt{Sign}$). It basically activates the $\mathtt{Send}(\mathsf{vk}, m, P)$ and $\mathtt{Rec}(\mathsf{vk}, m, 0)$ oracles.

– $\mathtt{Exec}^*(\mathsf{vk}, m, \sigma, P)$: This oracle outputs the transcript of an honest execution between a sender with private input P and the recipient with a valid signature σ (provided by the adversary). It basically activates the $\mathtt{Send}(\mathsf{vk}, m, P)$ and $\mathtt{Rec}^*(\mathsf{vk}, m, \sigma, 0)$ oracles.

Remark 2. The OSBE schemes proposed in [27] do not satisfy the semantic security w.r.t. the authority. This is obvious for the generic construction based on identity-based encryption which consists in only one flow of communication (since a scheme that achieves the strong security notions requires at least two flows). This is also true (to a lesser extent) for the RSA-based construction: for any third party, the semantic security relies (in the random oracle model) on the CDH assumption in a 2048-bit RSA group; but for the authority, it can be broken by solving two 1024-bit discrete logarithm problems. This task is much simpler in particular if the authority generates the RSA modulus $N = pq$ dishonestly (*e.g.* with $p - 1$ and $q - 1$ smooth). In order to make the scheme secure in our strong model, one needs (at least) to double the size of the RSA modulus and to make sure that the authority has selected and correctly employed a truly random seed in the generation of the RSA key pair [25].

3 An Efficient OSBE Scheme

In this section, we present a high-level instantiation of OSBE with the previous primitives as black boxes. Thereafter, we provide a specific instantiation with linear ciphertexts. The overall security then relies on the DLin assumption, a quite standard assumption in the standard model. Its efficiency is of the same order of magnitude than the construction based on identity-based encryption [27] (that only achieves weaker security notions) and better than the RSA-based scheme which provides similar security guarantees (in the random oracle model).

3.1 High-Level Instantiation

We assume we have an encryption scheme $\mathcal{E}$, a signature scheme $\mathcal{S}$ and a SPHF system onto a set $\mathbb{G}$. We additionally use a key derivation function KDF to derive a pseudo-random bit-string $K \in \{0,1\}^\ell$ from a pseudo-random element v in $\mathbb{G}$. One can use the Leftover-Hash Lemma [23], with a random seed defined in param during the global setup, to extract the entropy from v, then followed by a pseudo-random generator to get a long enough bit-string. Many uses of the same seed in the Leftover-Hash-Lemma just leads to a security loss linear in the number of extractions. We describe an oblivious signature-based envelope system $\mathcal{OSBE}$, to send a private message $P \in \{0,1\}^\ell$:

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter:
  - it first generates the global parameters for the signature scheme (using SSetup), the encryption scheme (using ESetup), and the SPHF system (using $\mathsf{SPHFSetup}$);
  - it then generates the public key ek of the encryption scheme (using $\mathsf{EKeyGen}$, while the decryption key will not be used);

  The output param consists of all the individual param and the encryption key ek;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$ runs $\mathsf{SKeyGen}(\mathsf{param})$ to generate a pair $(\mathsf{vk}, \mathsf{sk})$ of verification-signing keys;
- The $\mathsf{OSBESign}$ and $\mathsf{OSBEVerif}$ algorithms are exactly Sign and Verif from the signature scheme;
- $\mathsf{OSBEProtocol}\langle \mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$: In the following, $\mathcal{L} = \mathcal{L}(\mathsf{vk}, M)$ will describe the language of the ciphertexts under the above encryption key ek of a valid signature of the input message M under the input verification key vk (hence vk and M as inputs, while param contains ek).
  - $\mathcal{R}$ generates and sends $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
  - $\mathcal{S}$ computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$; $\mathcal{S}$ sends hp, Q to $\mathcal{R}$;
  - $\mathcal{R}$ computes $v' = \mathsf{ProjHash}(\mathsf{hp}, (\mathcal{L}, \mathsf{param}), c, r)$ and $P' = Q \oplus \mathsf{KDF}(v')$.

3.2 Security Properties

Theorem 3 (Correct). *$\mathcal{OSBE}$ is **sound**.*

Proof. Under the correctness of the SPHF system, $v' = v$, and thus $P' = (P \oplus \mathsf{KDF}(v)) \oplus \mathsf{KDF}(v') = P$.

Theorem 4 (Escrow-Free). *$\mathcal{OSBE}$ is **escrow-free** if the encryption scheme $\mathcal{E}$ is* semantically secure*: $\mathsf{Adv}^{\mathsf{esc}}_{\mathcal{OSBE}}(k, t) \le \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k, t')$ with $t' \approx t$.*

Proof. Let us assume $\mathcal{A}$ is an adversary against the escrow-free property of our scheme: The malicious adversary $\mathcal{A}$ is able to tell the difference between an interaction with $\mathcal{R}_0$ (who knows and uses a valid signature) and $\mathcal{R}_1$ (who does not use a valid signature), with advantage ε.

We now build an adversary $\mathcal{B}$ against the semantic security of the encryption scheme $\mathcal{E}$:

- $\mathcal{B}$ is first given the parameters for $\mathcal{E}$ and an encryption key ek;
- $\mathcal{B}$ emulates $\mathsf{OSBESetup}$: it runs SSetup and $\mathsf{SPHFSetup}$ by itself. For the encryption scheme $\mathcal{E}$, the parameters and the key have already been provided by the challenger of the encryption security game;

- $\mathcal{A}$ provides the verification key vk;
- $\mathcal{B}$ has to simulate all the oracles:
  - $\texttt{Send}(\mathsf{vk}, M, P)$, for a message M and a private input P: upon receiving c, one computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$. One sends back (hp, Q);
  - $\texttt{Rec}^*(\mathsf{vk}, M, \sigma, 0)$, for a message M and a valid signature σ: $\mathcal{B}$ outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
  - $\texttt{Exec}^*(\mathsf{vk}, M, \sigma, P)$: one first runs $\texttt{Rec}(\mathsf{vk}, M, \sigma, 0)$ to generate c, that is provided to $\texttt{Send}(\mathsf{vk}, M, P)$, to generate (hp, Q).
- At some point, $\mathcal{A}$ outputs a message M and a valid signature σ, and $\mathcal{B}$ has to simulate $\texttt{Rec}^*(\mathsf{vk}, M, \sigma, b)$: $\mathcal{B}$ sets $\sigma_0 \leftarrow \sigma$ and sets σ_1 as a random string. It sends (σ_0, σ_1) to the challenger of the semantic security of the encryption scheme and gets back c, an encryption of σ_β, for a random unknown bit β. It outputs c;
- $\mathcal{B}$ provides again access to the above oracles, and $\mathcal{A}$ outputs a bit b', that $\mathcal{B}$ forwards as its guess β' for the β involved in the semantic security game for $\mathcal{E}$.

Note that the above simulation perfectly emulates $\mathsf{Exp}^{\mathsf{esc}-\beta}_{\mathcal{OSBE},\mathcal{A}}(k)$ (since basically b is β, and b' is β'):

$$\varepsilon = \mathsf{Adv}^{\mathsf{esc}}_{\mathcal{OSBE},\mathcal{A}}(k) = \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E},\mathcal{B}}(k) \leq \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k, t).$$

Theorem 5 (Semantically Secure). *$\mathcal{OSBE}$ is **semantically secure** if the signature is* unforgeable*, the SPHF is* smooth *and the encryption scheme is* semantically secure *(and under the pseudo-randomness of the* KDF*):*

$$\mathsf{Adv}^{\mathsf{sem}}_{\mathcal{OSBE}}(k,t) \leq q_U\, \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k,t') + 2\,\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k,q_S,t'') + 2\,\mathsf{Adv}^{\mathsf{smooth}}_{\mathcal{SPHF}}(k)\ \textit{with}\ t', t'' \approx t.$$

In the above formula, q_U denotes the number of interactions the adversary has with the sender, and q_S the number of signing queries the adversary asked.

Proof. Let us assume $\mathcal{A}$ is an adversary against the semantic security of our scheme: The malicious adversary $\mathcal{A}$ is able to tell the difference between an interaction with $\mathcal{S}_0$ (who owns P_0) and $\mathcal{S}_1$ (who owns P_1), with advantage ε. We start from this initial security game, and make slight modifications to bound ε.

Game $\mathcal{G}_0$. Let us emulate this security game:

- $\mathcal{B}$ emulates the initialization of the system: it runs $\mathsf{OSBESetup}$ by itself, and then $\mathsf{OSBEKeyGen}$ to generate $(\mathsf{vk}, \mathsf{sk})$;
- $\mathcal{B}$ has to simulate all the oracles:
  - $\texttt{Sign}(\mathsf{vk}, M)$ and $\texttt{Sign}^*(\mathsf{vk}, M)$: it runs the corresponding algorithm by itself;
  - $\texttt{Send}(\mathsf{vk}, M, P)$, for a message M and a private input P: upon receiving c, one computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$. One sends back (hp, Q);

  - $\texttt{Rec}(\mathsf{vk}, M, 0)$, for a message M: $\mathcal{B}$ asks for a valid signature σ on M, computes and outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
  - $\texttt{Exec}(\mathsf{vk}, M, P)$: one simply first runs $\texttt{Rec}(\mathsf{vk}, M, 0)$ to generate c, that is provided to $\texttt{Send}(\mathsf{vk}, M, P)$, to generate (hp, Q).
- At some point, $\mathcal{A}$ outputs a message M and two inputs (P_0, P_1) to distinguish the sender, and $\mathcal{B}$ call back the above $\texttt{Send}(\mathsf{vk}, M, P_b)$ simulation to interact with $\mathcal{A}$;
- $\mathcal{B}$ provides again access to the above oracles, and $\mathcal{A}$ outputs a bit b'.

In this game, $\mathcal{A}$ has an advantage ε in guessing b:

$$\varepsilon = \Pr_{0}[b' = 1 | b = 1] - \Pr_{\mathcal{G}_0}[b' = 1 | b = 0] = 2 \times \Pr_{\mathcal{G}_0}[b' = b] - 1.$$

Game $\mathcal{G}_1^\beta$. This game involves the semantic security of the encryption scheme: $\mathcal{B}$ is already provided the parameters and the encryption key ek by the challenger of the semantic security of the encryption scheme, hence the initialization is slightly modified. In addition, $\mathcal{B}$ randomly chooses the bit b, and modifies the $\texttt{Rec}$ oracle simulation:

- $\texttt{Rec}(\mathsf{vk}, M, 0)$, for a message M: $\mathcal{B}$ asks for a valid signature σ_0 on M, and sets σ_1 as a random string, computes and outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma_b; r)$.

Since $\mathcal{B}$ knows b, it finally outputs $\beta' = (b' = b)$.

Note that $\mathcal{G}_1^0$ is exactly $\mathcal{G}_0$, and the distance between $\mathcal{G}_1^0$ and $\mathcal{G}_1^1$ relies on the Left-or-Right security of the encryption scheme, which can be shown equivalent to the semantic security, with a lost linear in the number of encryption queries, which is actually the number q_U of interactions with a user (the sender in this case), due to the hybrid argument [4]:

$$\begin{aligned} q_U \times \mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind}}(k) &\geq \Pr[\beta' = 1 | \beta = 0] - \Pr[\beta' = 1 | \beta = 1] \\ &= \Pr[b' = b | \beta = 0] - \Pr[b' = b | \beta = 1] \\ &= (2 \times \Pr_{\mathcal{G}_1^0}[b' = b] - 1) - (2 \times \Pr_{\mathcal{G}_1^1}[b' = b] - 1) \end{aligned}$$

As a consequence: $\varepsilon \leq q_U \times \mathsf{Adv}_{\mathcal{E}}^{\mathsf{ind}}(k) + (2 \times \Pr_{\mathcal{G}_1^1}[b' = b] - 1)$.

Game $\mathcal{G}_2$. This game involves the unforgeability of the signature scheme: $\mathcal{B}$ is already provided the parameters and the verification vk for the signature scheme, together with access to the signing oracle (note that all the signing queries Sign^* asked by the adversary in the $\texttt{FIND}$ stage, i.e., before the challenge interaction with $\texttt{Send}(\mathsf{vk}, M, P_b)$, are stored in $\mathcal{SM}$). The simulator $\mathcal{B}$ generates itself all the other parameters and keys, an namely the encryption key ek, together with the associated decryption key dk. For the $\texttt{Rec}$ oracle simulation, $\mathcal{B}$ keeps the random version (as in $\mathcal{G}_1^1$). In the challenge interaction with $\texttt{Send}(\mathsf{vk}, M, P_b)$, one stops the simulation and makes the adversary win if it uses a valid signature on a message $M \notin \mathcal{SM}$:

– $\mathtt{Send}(\mathsf{vk}, M, P_b)$, during the challenge interaction: upon receiving c, if $\mathcal{M} \notin \mathcal{SM}$, it first decrypts c to get the input signature σ. If σ is a valid signature, one stops the game, sets $b' = b$ and outputs b'. If the signature is in not valid, the simulation remains unchanged;
– $\mathtt{Rec}(\mathsf{vk}, M, 0)$, for a message M: $\mathcal{B}$ sets σ as a random string, computes and outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$.

Because of the abort in the case of a valid signature on a new message, we know that the adversary cannot use such a valid signature in the challenge. So, since M should not be in $\mathcal{SM}$, the signature will be invalid. Actually, the unique difference from the previous game $\mathcal{G}_1^1$ is the abort in case of valid signature on a new message in the challenge phase, which probability is bounded by $\mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, q_S)$. Using Shoup's Lemma [29]:

$$\Pr_{\mathcal{G}_1^1}[b' = b] - \Pr_{\mathcal{G}_2}[b' = b] \leq \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, q_S).$$

As a consequence: $\varepsilon \leq q_U \times \mathsf{Adv}^{\mathsf{ind}}_{\mathcal{E}}(k) + 2 \times \mathsf{Succ}^{\mathsf{euf}}_{\mathcal{S}}(k, q_S) + (2 \times \Pr_{\mathcal{G}_2}[b' = b] - 1)$.

Game $\mathcal{G}_3$. The last game involves the smoothness of the SPHF: The unique difference is in the computation of v in $\mathtt{Send}$ simulation, in the challenge phase only: $\mathcal{B}$ chooses a random $v \in \mathbb{G}$. Due to the statistical randomness of v in the previous game, in case the signature is not valid (a word that is not in the language), this game is statistically indistinguishable from the previous one:

$$\Pr_{\mathcal{G}_2}[b' = b] - \Pr_{\mathcal{G}_3}[b' = b] \leq \mathsf{Adv}^{\mathsf{smooth}}_{\mathcal{SPHF}}(k).$$

Since P_b is now masked by a truly random value, no information leaks on b: $\Pr_{\mathcal{G}_3}[b' = b] = 1/2$.

Theorem 6. *$\mathcal{OSBE}$ is **semantically secure w.r.t. the authority** if the SPHF is* pseudo-random *(and under the pseudo-randomness of the* KDF*):*

$$\mathsf{Adv}^{\mathsf{sem}^*}_{\mathcal{OSBE}}(k, t) \leq 2 \times \mathsf{Adv}^{\mathsf{pr}}_{\mathcal{SPHF}}(k, t).$$

Proof. Let us assume $\mathcal{A}$ is an adversary against the semantic security w.r.t. the authority: The malicious adversary $\mathcal{A}$ is able to tell the difference between an eavesdropped interaction with $\mathcal{S}_0$ (who owns P_0) and $\mathcal{S}_1$ (who owns P_1), with advantage ε. We start from this initial security game, and make slight modifications to bound ε.

Game $\mathcal{G}_0$. Let us emulate this security game:

– $\mathcal{B}$ emulates the initialization of the system: it runs $\mathsf{OSBESetup}$ by itself;
– $\mathcal{A}$ provides the verification key vk;
– $\mathcal{B}$ has to simulate all the oracles:
    - $\mathtt{Send}(\mathsf{vk}, M, P)$, for a message M and a private input P: upon receiving c, one computes $\mathsf{hk} = \mathsf{HashKG}(\mathcal{L}, \mathsf{param})$, $\mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, $v = \mathsf{Hash}(\mathsf{hk}, (\mathcal{L}, \mathsf{param}), c)$, and $Q = P \oplus \mathsf{KDF}(v)$. One sends back (hp, Q);

- $\mathtt{Rec}^*(\mathsf{vk}, M, \sigma, 0)$, for a message M and a valid signature σ: $\mathcal{B}$ outputs $c = \mathsf{Encrypt}(\mathsf{ek}, \sigma; r)$;
- $\mathtt{Exec}^*(\mathsf{vk}, M, \sigma, P)$: one first runs $\mathtt{Rec}(\mathsf{vk}, M, \sigma, 0)$ to generate c, that is provided to $\mathtt{Send}(\mathsf{vk}, M, P)$, to generate (hp, Q).

– At some point, $\mathcal{A}$ outputs a message M with a valid signature σ, and two inputs (P_0, P_1) to distinguish the sender, and $\mathcal{B}$ call back the above $\mathtt{Send}(\mathsf{vk}, M, P_b)$ and $\mathtt{Rec}^*(\mathsf{vk}, M, \sigma, 0)$ simulations to interact together and output the transcript $(c; \mathsf{hp}, Q)$;

– $\mathcal{B}$ provides again access to the above oracles, and $\mathcal{A}$ outputs a bit b'.

In this game, $\mathcal{A}$ has an advantage ε in guessing b:

$$\varepsilon = \Pr_{\mathcal{G}_0}[b' = 1|b = 1] - \Pr_{\mathcal{G}_0}[b' = 1|b = 0] = 2 \times \Pr_{\mathcal{G}_0}[b' = b] - 1.$$

Game $\mathcal{G}_1$. This game involves the pseudo-randomness of the SPHF: The unique difference is in the computation of v in $\mathtt{Send}$ simulation of the eavesdropped interaction, and so for the transcript: $\mathcal{B}$ chooses a random $v \in \mathbb{G}$ and computes $Q = P_b \oplus \mathsf{KDF}(v)$. Due to the pseudo-randomness of v in the previous game, since $\mathcal{A}$ does not know the random coins r used to encrypt σ, this game is computationally indistinguishable from the previous one.

$$\Pr_{\mathcal{G}_1}[b' = b] - \Pr_{\mathcal{G}_0}[b' = b] \leq \mathsf{Adv}^{\mathsf{pr}}_{\mathcal{SPHF}}(k, t).$$

Since P_b is now masked by a truly random value v, no information leaks on b: $\Pr_{\mathcal{G}_1}[b' = b] = 1/2$.

3.3 Our Efficient OSBE Instantiation

Our first construction combines the linear encryption scheme [7], the Waters signature scheme [30] and a SPHF on linear ciphertexts [13,28]. It thus relies on classical assumptions: CDH for the unforgeability of signatures and DLin for the semantic security of the encryption scheme. The formal definitions are recalled in the full version [6].

Basic Primitives. Given an encrypted Waters signature from the recipient, the sender is able to compute a projection key, and a hash corresponding to the expected signature, and send to the recipient the projection key and the product between the expected hash and the message P. If the recipient was honest (a correct ciphertext), it is able to compute the hash thanks to the projection key, and so to find P, in the other case it does not learn anything.

We briefly sketch the basic building blocks: linear encryption, Waters signature and the SPHF for linear tuples.

All these primitives work in a pairing-friendly environment $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, 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.

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 $\mathcal{F}(M) = u_0 \prod_{i=1}^{k} u_i^{M_i}$. The public verification key is $\mathsf{vk} = g^z$, which corresponding secret signing key is $\mathsf{sk} = h^z$, for a random $z \xleftarrow{\$} \mathbb{Z}_p$. The signature on a message $M \in \{0,1\}^k$ is $\sigma = \big(\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^s, \sigma_2 = g^s\big)$, for some random $s \xleftarrow{\$} \mathbb{Z}_p$. It can be verified by checking $e(g, \sigma_1) = e(\mathsf{vk}, h) \cdot e(\mathcal{F}(M), \sigma_2)$. This signature scheme is *unforgeable* under the CDH assumption.

Linear Encryption. The secret key dk is a pair of random scalars (y_1, y_2) and the public key is $\mathsf{ek} = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$. One encrypts a message $M \in \mathbb{G}$ as $c = \big(c_1 = Y_1^{r_1}, c_2 = Y_2^{r_2}, c_3 = g^{r_1+r_2} \cdot M\big)$, for random scalars $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$. To decrypt, one computes $M = c_3/(c_1^{1/y_1} c_2^{1/y_2})$. This encryption scheme is *semantically secure* under the DLin assumption.

DLin*-compatible Smooth-Projective Hash Function.* This is actually a weaker variant of [28]. The language $\mathcal{L}$ consists of the linear tuples w.r.t. a basis (u, v, g). For a linear encryption key $\mathsf{ek} = (Y_1, Y_2)$, a ciphertext $C = (c_1, c_2, c_3)$ is an encryption of the message M if $(c_1, c_2, c_3/M)$ is a linear tuple w.r.t. the basis (Y_1, Y_2, g). The language $\mathsf{Lin}(\mathsf{ek}, M)$ consists of these ciphertexts. An SPHF for this language can be:

$$\begin{aligned}
\mathsf{HashKG}(\mathsf{Lin}(\mathsf{ek}, M)) &= \mathsf{hk} = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3 \\
\mathsf{Hash}(\mathsf{hk}; \mathsf{Lin}(\mathsf{ek}, M), C) &= c_1^{x_1} c_2^{x_2} (c_3/M)^{x_3} \\
\mathsf{ProjKG}(\mathsf{hk}; \mathsf{Lin}(\mathsf{ek}, M), C) &= \mathsf{hp} = (Y_1^{x_1} g^{x_3}, Y_2^{x_2} g^{x_3}) \\
\mathsf{ProjHash}(\mathsf{hp}; \mathsf{Lin}(\mathsf{ek}, M), C; r) &= \mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}
\end{aligned}$$

This function is defined for linear tuples in $\mathbb{G}$, but it could work in any group, since it does not make use of pairings. And namely, we use it below in $\mathbb{G}_T$.

Smooth-Projective Hash Function for Linear Encryption of Valid Waters Signatures. We will consider a slightly more complex language: the ciphertexts under ek of a valid signature of M under vk. A given ciphertext $C = (c_1, c_2, c_3, \sigma_2)$ contains a valid signature of M if and only if (c_1, c_2, c_3) actually encrypts σ_1 such that (σ_1, σ_2) is a valid Waters signature on M. The latter means

$$(C_1 = e(c_1, g), C_2 = e(c_2, g), C_3 = e(c_3, g)/(e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2))$$

is a linear tuple in basis $(U = e(Y_1, g), V = e(Y_2, g), g_t = e(g, g))$ in $\mathbb{G}_T$. Since the basis consists of 3 elements of the form $e(\cdot, g)$, the projected key can be compacted in $\mathbb{G}$. We thus consider the language $\mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M)$ that contains these quadruples $(c_1, c_2, c_3, \sigma_2)$, and its SPHF:

$$\begin{aligned}
\mathsf{HashKG}(\mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M)) &= \mathsf{hk} = (x_1, x_2, x_3) \xleftarrow{\$} \mathbb{Z}_p^3 \\
\mathsf{Hash}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) &= \\
&\qquad e(c_1, g)^{x_1} e(c_2, g)^{x_2} (e(c_3, g)/(e(h, \mathsf{vk}) e(\mathcal{F}(M), \sigma_2)))^{x_3} \\
\mathsf{ProjKG}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) &= \mathsf{hp} = (\mathsf{ek}_1^{x_1} g^{x_3}, \mathsf{ek}_2^{x_2} g^{x_3}) \\
\mathsf{ProjHash}(\mathsf{hp}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C; r) &= e(\mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}, g)
\end{aligned}$$

Instantiation. We now define our OSBE protocol, where a sender $\mathcal{S}$ wants to send a private message $P \in \{0,1\}^\ell$ to a recipient $\mathcal{R}$ in possession of a Waters signature on a message M.

- $\mathsf{OSBESetup}(1^k)$, where k is the security parameter, defines a pairing-friendly environment $(p, \mathbb{G}, g, \mathbb{G}_T, e)$, the public parameters $h \xleftarrow{\$} \mathbb{G}$, an encryption key $\mathsf{ek} = (Y_1 = g^{y_1}, Y_2 = g^{y_2})$, where $(y_1, y_2) \xleftarrow{\$} \mathbb{Z}_p^2$, and $\boldsymbol{u} = (u_0, \ldots, u_k) \xleftarrow{\$} \mathbb{G}^{k+1}$ for the Waters signature. All these elements constitute the string param;
- $\mathsf{OSBEKeyGen}(\mathsf{param})$, the authority generates a pair of keys $(\mathsf{vk} = g^z, \mathsf{sk} = h^z)$ for a random scalar $z \xleftarrow{\$} \mathbb{Z}_p$;
- $\mathsf{OSBESign}(\mathsf{sk}, M)$ produces a signature $\sigma = (h^z \mathcal{F}(M)^s, g^s)$;
- $\mathsf{OSBEVerif}(\mathsf{vk}, M, \sigma)$ checks if $e(\sigma_1, g) = e(\sigma_2, \mathcal{F}(M)) \cdot e(h, \mathsf{vk})$.
- $\mathsf{OSBEProtocol}\langle \mathcal{S}(\mathsf{vk}, M, P), \mathcal{R}(\mathsf{vk}, M, \sigma)\rangle$ runs as follows:
  - $\mathcal{R}$ chooses random $r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$ and sends a linear encryption of σ:
    $C = (c_1 = \mathsf{ek}_1^{r_1}, c_2 = \mathsf{ek}_2^{r_2}, c_3 = g^{r_1+r_2} \cdot \sigma_1, \sigma_2)$
  - $\mathcal{S}$ chooses random $x_1, x_2, x_3 \xleftarrow{\$} \mathbb{Z}_p^3$ and computes:
    * $\mathsf{HashKG}(\mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M)) = \mathsf{hk} = (x_1, x_2, x_3)$;
    * $\mathsf{Hash}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) = v =$
      $e(c_1, g)^{x_1} e(c_2, g)^{x_2} (e(c_3, g)/(e(h, \mathsf{vk}) e(\mathcal{F}(M), \sigma_2)))^{x_3}$;
    * $\mathsf{ProjKG}(\mathsf{hk}; \mathsf{WLin}(\mathsf{ek}, \mathsf{vk}, M), C) = \mathsf{hp} = (\mathsf{ek}_1^{x_1} g^{x_3}, \mathsf{ek}_2^{x_2} g^{x_3})$.
  - $\mathcal{S}$ then sends $(\mathsf{hp}, Q = P \oplus \mathsf{KDF}(v))$ to $\mathcal{R}$;
  - $\mathcal{R}$ computes $v' = e(\mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}, g)$ and $P' = Q \oplus \mathsf{KDF}(v')$.

An asymmetric instantiation can be found in the full version [6].

3.4 Security and Efficiency

We now provide a security analysis of this scheme. This instantiation differs, from the high-level instantiation presented before, in the ciphertext C of the signature $\sigma = (\sigma_1, \sigma_2)$. The second half of the signature indeed remains in clear. It thus does not guarantee the semantic security on the signature used in the ciphertext. However, granted Waters signature randomizability, one can re-randomize the signature each time, and thus provide a totally new σ_2: it does not leak any information about the original signature. The first part of the ciphertext (c_1, c_2, c_3) does not leak any additional information under the DLin assumption. As a consequence, the global ciphertext guarantees the semantic security of the original signature if a new re-randomized signature is encrypted each time. We can now apply the high-level construction security, and all the assumptions hold under the DLin one:

Theorem 7. *Our OSBE scheme is secure (i.e., escrow-free, semantically secure, and semantically secure w.r.t. the authority) under the* DLin *assumption (and the pseudo-random generator in the* KDF*).*

Our proposed scheme needs one communication for $\mathcal{R}$ and one for $\mathcal{S}$, so it is round-optimal. Communication also consists of few elements, $\mathcal{R}$ sends 4 group elements, and $\mathcal{S}$ answers with 2 group elements only and an ℓ-bit string for the masked $P \in \{0,1\}^{\ell}$. As explained in Remark 2, this has to be compared with the RSA-based scheme from [27] which requires 2 elements in RSA groups (with double-length modulus). For a 128-bit security level, using standard Type-I bilinear groups implementation [16], we obtain a 62.5% improvement[1] in communication complexity over the RSA-based scheme proposed in the original paper [27].

While reducing the communication cost of the scheme, we have improved its security and it now fits the proposed applications. In [27], such schemes were proposed for applications where someone wants to transmit a confidential information to an agent belonging to a specific agency. However the agent does not want to give away his signature. As they do not consider eavesdropping and replay in their semantic security nothing prevents an adversary to replay a part of a previous interaction to impersonate a CIA agent (to recall their example). In practice, an additional secure communication channel, such as with SSL, was required in their security model, hence increasing the communication cost: our protocol is secure by itself.

4 An Efficient Blind Signature

4.1 Definitions

A more formal definition of blind signatures is provided in the full version [6], but we briefly recall it in this section: A blind signature scheme $\mathcal{BS}$ is defined by a setup algorithm $\mathsf{BSSetup}(1^k)$ that generates the global parameters param, and key generation algorithm $\mathsf{BSKeyGen}(\mathsf{param})$ that outputs a pair $(\mathsf{vk}, \mathsf{sk})$, and interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$ which provides $\mathcal{U}$ with a signature on m, and a verification algorithm $\mathsf{BSVerif}(\mathsf{vk}, m, \sigma)$ that checks its validity. The security of a blind signature scheme is defined through the unforgeability and blindness properties: An adversary against the unforgeability tries to generate $q_s + 1$ valid message-signature pairs after at most q_s complete interactions with the honest signer; The blindness condition states that a malicious signer should be unable to decide which of two messages m_0, m_1 has been signed first in two executions with an honest user.

4.2 Our Instantiation

We now present a new way to obtain a blind signature scheme in the standard model under classical assumptions with a common-reference string. This is an improvement over [5]. We are going to use the same building blocks as before, so linear encryption, Waters signatures and a SPHF on linear ciphertexts. More elaborated languages will be required, but just conjunctions and disjunctions of

[1] The improvement is even more important for the scheme described in the full version where the size drops down to 3/16-th.

classical languages, as done in [1] (see the full version [6]), hence the efficient construction. Our blind signature scheme is defined by:

- $\mathsf{BSSetup}(1^k)$, generates a pairing-friendly system $(p, \mathbb{G}, g, \mathbb{G}_T, e)$ and an encryption key $\mathsf{ek} = (u, v, g) \in \mathbb{G}^3$. It also chooses at random $h \in \mathbb{G}$ and generators $\boldsymbol{u} = (u_i)_{i \in [\![1,\ell]\!]} \in \mathbb{G}^\ell$ for the Waters function. It outputs the global parameters $\mathsf{param} = (p, \mathbb{G}, g, \mathbb{G}_T, e, \mathsf{ek}, h, \boldsymbol{u})$;
- $\mathsf{BSKeyGen}(\mathsf{param})$ picks at random a secret key $\mathsf{sk} = x$ and computes the verification key $\mathsf{vk} = g^x$;
- $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, m)\rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on M
  - $\mathcal{U}$ computes the bit-per-bit encryption of M by encrypting each $u_i^{M_i}$ in b_i, $\forall i \in [\![1,\ell]\!], b_i = \mathsf{Encrypt}(\mathsf{ek}, u_i^{M_i}; (r_{i,1}, r_{i,2})) = (u^{r_{i,1}}, v^{r_{i,2}}, g^{r_{i,1}+r_{i,2}} u_i^{M_i})$. Then writing $r_1 = \sum r_{i,1}$ and $r_2 = \sum r_{i,2}$, he computes the encryption c of $\mathsf{vk}^{r_1+r_2}$ with $\mathsf{Encrypt}(\mathsf{ek}, \mathsf{vk}^{r_1+r_2}; (s_1, s_2)) = (u^{s_1}, v^{s_2}, g^{s_1+s_2}\mathsf{vk}^{r_1+r_2})$. $\mathcal{U}$ then sends $(c, (b_i))$;
  - On input of these ciphertexts, the algorithm $\mathcal{S}$ computes the corresponding SPHF, considering the language $\mathcal{L}$ of valid ciphertexts. This is the conjunction of several languages :
    1. One checking that each b_i encrypts a bit in basis u_i: in $\mathsf{BLin}(\mathsf{ek}, u_i)$;
    2. One considering $(d_1, d_2, c_1, c_2, c_3)$, that checks if (c_1, c_2, c_3) encrypts an element d_3 such that (d_1, d_2, d_3) is a linear tuple in basis (u, v, vk): in $\mathsf{ELin}(\mathsf{ek}, \mathsf{vk})$, where $d_1 = \prod_i b_{i,1}$ and $d_2 = \prod_i b_{i,2}$.
  - $\mathcal{S}$ computes the corresponding Hash-value v, extracts $K = \mathsf{KDF}(v) \in \mathbb{Z}_p$, generates the blinded signature $(\sigma_1'' = h^x \delta^s, \sigma_2' = g^s)$, where $\delta = u_0 \prod_i b_{i,3} = \mathcal{F}(M) g^{r_1+r_2}$, and sends $(\mathsf{hp}, Q = \sigma_1'' \times g^K, \sigma_2')$;
  - Upon receiving $(\mathsf{hp}, Q, \sigma_2')$, using its witnesses and hp, $\mathcal{U}$ computes the $\mathsf{ProjHash}$-value v', extracts $K' = \mathsf{KDF}(v')$ and unmasks $\sigma_1'' = Q \times g^{-K'}$. Thanks to the knowledge of r_1 and r_2, it can compute $\sigma_1' = \sigma_1'' \times (\sigma_2')^{-r_1-r_2}$. Note that if $v' = v$, then $\sigma_1' = h^x \mathcal{F}(M)^s$, which together with $\sigma_2' = g^s$ is a valid Waters signature on M. It can thereafter re-randomize the final signature $\sigma = (\sigma_1' \cdot \mathcal{F}(M)^{s'}, \sigma_2' \cdot g^{s'})$.
- $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$, checks whether $e(\sigma_1, g) = e(h, \mathsf{vk}) \cdot e(\mathcal{F}(M), \sigma_2)$.

The idea is to remove any kind of proof of knowledge in the protocol, which was the main concern in [5], and use instead a SPHF. This way, we obtain a protocol where the user first sends $3\ell + 6$ group elements for the ciphertext, and receives back $5\ell + 4$ elements for the projection key and 2 group elements for the blinded signature. So $8\ell + 12$ group elements are used in total. This has to be compared to $9\ell + 24$ in [5]. We both reduce the linear and the constant parts in the number of group elements involved while relying on the same hypotheses. And the final result is still a standard Waters signature.

Remark 8. In [17], Garg *el al.* proposed the first round-optimal blind signature scheme in the standard model, without CRS. In order to remove the CRS, their scheme makes use of ZAPs [14] and is quite inefficient. Moreover, its security relies on a stronger assumption (namely, sub-exponential hardness of one-to-one

one-way functions). A natural idea is to replace the CRS in our scheme with Groth-Ostrovsky-Sahai ZAP [21] based on the DLin assumption. This change would only double the communication complexity, but we do not know how to prove the security of the resulting scheme[2]. It remains a tantalizing open problem to design an efficient round-optimal blind signature in the standard model without CRS.

4.3 Security

In blind signatures, one expects two kinds of security properties:

– *blindness*, preventing the signer to be able to recognize which message was signed during a specific interaction. Due to Waters re-randomizability and linear encryption, this property is guaranteed in our scheme under the DLin assumption;
– *unforgeability*, guaranteeing the user will not be able to output more signed messages than the number of actual interactions. In this scheme, granted the extractability of the encryption (the simulator can know the decryption key) one can show that the user cannot provide a signature on a message different from the ones it asked to be blindly signed. Hence, the unforgeability relies on the Waters unforgeability, that is the CDH assumption.

Theorem 9. *Our blind signature scheme is blind*[3] *under the* DLin *assumption (and the pseudo-randomness of the* KDF*) and unforgeable under the* CDH *assumption.*

A full proof can be found in the full version [6].

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.

References

1. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth Projective Hashing for Conditionally Extractable Commitments. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 671–689. Springer, Heidelberg (2009)
2. 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)

[2] Indeed, opening the commitment scheme in the ZAP and forging a signature relies on the same computational assumption, which makes it impossible to apply the complexity leveraging argument from [17].

[3] Our scheme satisfies the *a posteriori blindness* security notion – introduced in [24] – that models exactly what the signer sees in the real world. As mentioned in [24], it formalizes the security desired for most applications of blind signatures (*e.g.* e-cash or e-voting).

3. Balfanz, D., Durfee, G., Shankar, N., Smetters, D., Staddon, J., Wong, H.-C.: Secret handshakes from pairing-based key agreements. In: IEEE Symposium on Security and Privacy, pp. 180–196 (2003)
4. Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A concrete security treatment of symmetric encryption. In: 38th Annual Symposium on Foundations of Computer Science, pp. 394–403. IEEE Computer Society Press (October 1997)
5. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on Randomizable Ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg (2011)
6. Blazy, O., Pointcheval, D., Vergnaud, D.: Round-Optimal Privacy-Preserving Protocols with Smooth Projective Hash Functions. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 94–110. Springer, Heidelberg (2012)
7. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
8. 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)
9. Boyen, X., Waters, B.: Full-Domain Subgroup Hiding and Constant-Size Group Signatures. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 1–15. Springer, Heidelberg (2007)
10. Bradshaw, R.W., Holt, J.E., Seamons, K.E.: Concealing complex policies with hidden credentials. In: Atluri, V., Pfitzmann, B., McDaniel, P. (eds.) ACM CCS 2004: 11th Conference on Computer and Communications Security, pp. 146–157. ACM Press (October 2004)
11. Castelluccia, C., Jarecki, S., Tsudik, G.: Secret Handshakes from CA-Oblivious Encryption. In: Lee, P.J. (ed.) ASIACRYPT 2004. LNCS, vol. 3329, pp. 293–307. Springer, Heidelberg (2004)
12. 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)
13. Cramer, R., Shoup, V.: Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (2002)
14. Dwork, C., Naor, M.: Zaps and their applications. SIAM J. Comput. 36(6), 1513–1543 (2007)
15. 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)
16. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)
17. Garg, S., Rao, V., Sahai, A., Schröder, D., Unruh, D.: Round Optimal Blind Signatures. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 630–648. Springer, Heidelberg (2011)
18. Gennaro, R., Lindell, Y.: A Framework for Password-Based Authenticated Key Exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (2003), `http://eprint.iacr.org/2003/032.ps.gz`
19. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. ACM Transactions on Information and System Security 9(2), 181–234 (2006)

20. Groth, J.: Fully Anonymous Group Signatures Without Random Oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
21. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive Zaps and New Techniques for NIZK. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
22. 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)
23. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM Journal on Computing 28(4), 1364–1396 (1999)
24. Hazay, C., Katz, J., Koo, C.-Y., Lindell, Y.: Concurrently-Secure Blind Signatures Without Random Oracles or Setup Assumptions. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 323–341. Springer, Heidelberg (2007)
25. Juels, A., Guajardo, J.: RSA Key Generation with Verifiable Randomness. In: Naccache, D., Paillier, P. (eds.) PKC 2002. LNCS, vol. 2274, pp. 357–374. Springer, Heidelberg (2002)
26. Kalai, Y.T.: Smooth Projective Hashing and Two-Message Oblivious Transfer. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 78–95. Springer, Heidelberg (2005)
27. Li, N., Du, W., Boneh, D.: Oblivious signature-based envelope. In: 22nd ACM Symposium Annual on Principles of Distributed Computing, pp. 182–189. ACM Press (July 2003)
28. Shacham, H.: A Cramer-Shoup encryption scheme from the Linear Assumption and from progressively weaker Linear variants. Cryptology ePrint Archive, Report 2007/074 (February 2007), `http://eprint.iacr.org/`
29. Shoup, V.: OAEP reconsidered. Journal of Cryptology 15(4), 223–249 (2002)
30. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)