New Techniques for SPHFs and Efficient One-Round PAKE Protocols

Fabrice Benhamouda[1], Olivier Blazy[2], Céline Chevalier[3], David Pointcheval[1], and Damien Vergnaud[1]

[1] ENS, Paris, France †
[2] Ruhr-Universität Bochum, Germany
[3] Université Panthéon–Assas, Paris, France

Abstract. *Password-authenticated key exchange* (PAKE) protocols allow two players to agree on a shared high entropy secret key, that depends on their own passwords only. Following the Gennaro and Lindell's approach, with a new kind of *smooth-projective hash functions* (SPHFs), Katz and Vaikuntanathan recently came up with the first concrete one-round PAKE protocols, where the two players just have to send simultaneous flows to each other. The first one is secure in the Bellare-Pointcheval-Rogaway (BPR) model and the second one in the Canetti's UC framework, but at the cost of *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proofs (one for the BPR-secure protocol and two for the UC-secure one), which make the overall constructions not really efficient.

This paper follows their path with, first, a new efficient instantiation of SPHF on Cramer-Shoup ciphertexts, which allows to get rid of the SS-NIZK proof and leads to the design of the most efficient one-round PAKE known so far, in the BPR model, and in addition without pairings. In the UC framework, the security proof required the simulator to be able to extract the hashing key of the SPHF, hence the additional SS-NIZK proof. We improve the way the latter extractability is obtained by introducing the notion of *trapdoor smooth projective hash functions* (TSPHFs). Our concrete instantiation leads to the most efficient one-round PAKE UC-secure against static corruptions to date.

We additionally show how these SPHFs and TSPHFs can be used for blind signatures and zero-knowledge proofs with straight-line extractability.

1 Introduction

Authenticated Key Exchange protocols are quite important primitives for practical applications, since they enable two parties to generate a shared high entropy secret key, to be later used with symmetric primitives in order to protect communications, while interacting over an insecure network under the control of an adversary. Various authentication means have been proposed, and the most practical one is definitely a shared low entropy secret, or a password they can

† CNRS – UMR 8548 and INRIA – EPI Cascade

agree on over the phone, hence PAKE, for *Password-Authenticated Key Exchange*. The most famous instantiation has been proposed by Bellovin and Merritt [4], EKE for Encrypted Key Exchange, which simply consists of a Diffie-Hellman key exchange [17], where the flows are symmetrically encrypted under the shared password. Overall, the equivalent of 2 group elements have to be sent.

A first formal security model was proposed by Bellare, Pointcheval and Rogaway [3] (the BPR model), to deal with off-line dictionary attacks. It essentially says that the best attack should be the on-line exhaustive search, consisting in trying all the passwords by successive executions of the protocol with the server. Several variants of EKE with BPR-security proofs have been proposed in the ideal-cipher model or the random-oracle model [27]. Katz, Ostrovsky and Yung [23] proposed the first practical scheme (KOY), provably secure in the standard model under the DDH assumption. This is a 3-flow protocol, with the client sending 5 group elements plus a verification key and a signature, for a one-time signature scheme, and the server sending 5 group elements. It has been generalized by Gennaro and Lindell [20] (GL), making use of smooth projective hash functions.

Smooth Projective Hash Functions (SPHFs) were introduced by Cramer and Shoup [16] in order to achieve IND-CCA security from IND-CPA encryption schemes, which led to the first efficient IND-CCA encryption scheme provably secure in the standard model under the DDH assumption [15]. They can be seen as a kind of implicit designated-verifier proofs of membership [1, 9]. Basically, SPHFs are families of pairs of functions $(\mathsf{Hash}, \mathsf{ProjHash})$ defined on a language L. These functions are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where hk, the hashing key, can be seen as the private key and hp, the projection key, as the public key. On a word $W \in \mathrm{L}$, both functions should lead to the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ with the hashing key and $\mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, W, w)$ with the projection key only but also a witness w that $W \in \mathrm{L}$. Of course, if $W \notin \mathrm{L}$, such a witness does not exist, and the smoothness property states that $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ is independent of hp. As a consequence, even knowing hp, one cannot guess $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$.

One-Round PAKE in the BPR Model. Gennaro and Lindell [20] (GL) extended the initial definition of smooth projective hash functions for an application to PAKE. Their approach has thereafter been adapted to the *Universal Composability* (UC) framework by Canetti *et al.* [14], but for static corruptions only. It has been improved by Abdalla, Chevalier and Pointcheval [1] to resist to adaptive adversaries. But the 3-flow KOY protocol remains the most efficient protocol BPR-secure under the DDH assumption.

More recently, the ultimate step for PAKE has been achieved by Katz and Vaikuntanathan [24] (KV), who proposed a *practical* one-round PAKE, where the two players just have to send simultaneous flows to each other, that depend on their own passwords only. More precisely, each flow just consists of an IND-CCA ciphertext of the password and an SPHF projection key for the correctness of the partner's ciphertext (the word is the ciphertext and the witness consists of the random coins of the encryption). The shared secret key is eventually the product of the two hash values, as in the KOY and GL protocols.

Katz and Vaikuntanathan Smooth Projective Hash Functions. Because of the simultaneous flows, one flow cannot explicitly depend on the partner's flow, which makes impossible the use of the Gennaro and Lindell SPHF (later named GL-SPHF), in which the projection key depends on the word (the ciphertext here). On the other hand, the adversary can wait for the player to send his flow first, and then adapt its message, which requires stronger security notions than the initial Cramer and Shoup SPHF (later named CS-SPHF), in which the smoothness does not hold anymore if the word is generated after having seen the projection key. This led Katz and Vaikuntanathan to provide a new definition for SPHF (later named KV-SPHF), where the projection key depends on the hashing key only, and the smoothness holds even if the word is chosen after having seen the projection key. Variations between CS-SPHF, GL-SPHF and KV-SPHF are in the way one computes the projection key hp from the hashing key hk and the word W, but also in the smoothness property, according to the freedom the adversary has to choose W, when trying to distinguish the hash value from a random value. As a side note, while CS-SPHF is close to the initial definition, useful for converting an IND-CPA encryption scheme to IND-CCA, GL-SPHFs and KV-SPHFs did prove quite useful too: we will use KV-SPHFs for our one-round PAKE protocols and a GL-SPHF for the blind signature scheme.

As just explained, the strongest definition of SPHF, which gives a lot of freedom to the adversary, is the recent KV-SPHF. However, previous SPHFs known on Cramer-Shoup ciphertexts were GL-SPHFs only. For their one-round PAKE, Katz and Vaikuntanathan did not manage to construct such a KV-SPHF for an efficient IND-CCA encryption scheme. They then suggested to use the Naor and Yung approach [26], with an ElGamal-like encryption scheme and a *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proof [28]. Such an SS-NIZK proof is quite costly in general. They suggested to use Groth-Sahai [21] proofs in bilinear groups and the linear encryption [10] which leads to a PAKE secure under the DLin assumption with a ciphertext consisting of 66 group elements and a projection key consisting of 4 group elements. As a consequence, the two players have to send 70 group elements each, which is far more costly than the KOY protocol, but it is one-round only.

More recent results on SS-NIZK proofs or IND-CCA encryption schemes, in the discrete logarithm setting, improved on that: Libert and Yung [25] proposed a more efficient SS-NIZK proof of plaintext equality in the Naor-Yung-type cryptosystem with ElGamal-like encryption. The proof can be reduced from 60 to 22 group elements and the communication complexity of the resulting PAKE is decreased to 32 group elements per user. Jutla and Roy [22] proposed relatively-sound NIZK proofs as an efficient alternative to SS-NIZK proofs to build new publicly-verifiable IND-CCA encryption schemes. They can then decrease the PAKE communication complexity to 20 group elements per user. In any case, one can remark that all one-round PAKE schemes require pairing computations.

One-Round PAKE in the Universal Composability Framework. Katz and Vaikuntanathan [24] also proposed another construction of one-round PAKE, provably secure against static corruptions in the UC framework. To achieve such

a level of security, the simulator has to be more powerful: it should be able to make a successful execution after a dummy simulation, with a wrong password. To this aim, Katz and Vaikuntanathan allowed the simulator to extract the hashing key of the SPHF, to allow it to compute afterwards the hash value on any word, even outside the language. More precisely, each player additionally encrypts his hashing key to allow the key recovery by the simulator, so that the latter can compute the hash value even when a dummy password has initially been committed, whereas a success is expected. While this is the first one-round PAKE provably secure in the UC framework, hashing key recovery requires an additional quite costly simulation-sound extractable NIZK proof. Although the latter can also be improved by the above more recent work [22], the UC-secure one-round PAKE is still much more costly than the BPR-secure protocol.

Achievements. Our first contribution is the description of an instantiation of KV-SPHF on Cramer-Shoup ciphertexts, and thus the first KV-SPHF on an efficient IND-CCA encryption scheme. We thereafter use it within the above KV framework for one-round PAKE [24], in the BPR security model. Our scheme just consists of 6 group elements in each direction under the DDH assumption (4 for the ciphertext, and 2 for the projection key). This has to be compared with the 20 group elements, or more, in the best constructions discussed above, which all need pairing-friendly groups and pairing computations, or with the KOY protocol that has a similar complexity but with three sequential flows.

We also present the first GL-SPHFs/KV-SPHFs able to handle multi-exponentiation equations without requiring pairings. Those SPHFs are thus quite efficient. They lead to two applications. First, our new KV-SPHFs enable several efficient instantiations of one-round *Language-Authenticated Key-Exchange* (LAKE) protocols [5]. Our above one-round PAKE scheme is actually a particular case of a more general one-round LAKE scheme, for which we provide a BPR-like security model and a security proof. Our general constructions also cover Credential-Authenticated Key Exchange [11]. Second, thanks to a new GL-SPHF, we improve on the *blind signature* scheme presented in [9], from $5\ell + 6$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$ to $3\ell + 7$ group elements in $\mathbb{G}_1$ and 1 group element in $\mathbb{G}_2$, for an ℓ-bit message to be blindly signed with a Waters signature [29]. Our protocol is round-optimal, since it consists of two flows, and leads to a classical short Waters signature.

Our second contribution is the novel extension of SPHFs, called Trapdoor SPHFs, or TSPHFs. In addition to showing that an SPHF with an encryption of the hashing key and a simulation-sound extractable NIZK proof, as used in the UC-secure one-round PAKE of Katz and Vaikuntanathan, can be seen as an inefficient TSPHF, we provide efficient instantiations of TSPHFs. To do so, we first describe a new generic framework for SPHFs that allows an easy conversion to TSPHFs. We then apply it to our above KV-SPHF on a Cramer-Shoup ciphertext. Using our new TSPHF in the UC-secure one-round PAKE framework from [24], we obtain a scheme which consists of 11 group elements in each direction (actually, 6 group elements in $\mathbb{G}_1$ and 5 group elements in $\mathbb{G}_2$ in an asymmetric bilinear setting, using the Cramer-Shoup encryption). It is secure in

the UC framework against static corruptions under the SXDH assumption with a CRS, and just twice more costly than our above BPR-secure PAKE. This is the most efficient UC-secure one-round PAKE.

Finally, while SPHFs are often used as implicit designated-verifier proofs of membership, when one wants to make them explicit, by sending the hash value, one does not get the zero-knowledge property. We then show that TSPHFs actually lead to extractable zero-knowledge (E-ZK) arguments.

Outline. In Section 2, we first revisit the different definitions for SPHFs proposed in [16, 20, 24], respectively denoted CS-SPHFs, GL-SPHFs and KV-SPHFs, and give the first instantiation of KV-SPHF on Cramer-Shoup ciphertexts. This leads to our efficient one-round PAKE provably secure in the BPR model.

We then define our novel extension of SPHFs, called Trapdoor SPHFs, or TSPHFs, in Section 3. After the presentation of a new framework for SPHFs together with a generic way to convert SPHFs into TSPHFs, we provide efficient instantiations of TSPHFs in Section 4, and especially on Cramer-Shoup ciphertexts, which lead to our efficient UC-secure one-round PAKE scheme.

We conclude with more constructions in Section 5 and other applications of both SPHFs and TSPHFs: First, thanks to the various complex languages we can handle with SPHFs, in Section 6, we present our one-round LAKE and an improved *blind signature* scheme. Finally, we provide another application of our TSPHF constructions by presenting efficient E-ZK protocols in Section 7.

Full Versions. This paper was formed by merging two Crypto 2013 submissions, both extending SPHFs with applications to PAKE protocols. The first one provided more evolved SPHFs with the BPR-secure PAKE as an application, and the second introduced TSPHFs with application to UC-secure PAKE. Because of lack of space, many details are left to the full versions of both papers that are referred to along this paper as the SPHF full version [6] and the TSPHF full version [7] respectively.

2 New SPHF on Cramer-Shoup Ciphertexts

In this section, we first recall the definitions of SPHFs and present our classification based on the dependence between words and keys. According to this classification, there are three types of SPHFs: the (almost) initial Cramer and Shoup [16] type (CS-SPHF) introduced for enhancing an IND-CPA encryption scheme to IND-CCA, the Gennaro and Lindell [20] type (GL-SPHF) introduced for PAKE, and the Katz and Vaikuntanathan [24] type (KV-SPHF) introduced for one-round PAKE.

Then, after a quick review on the Cramer-Shoup encryption scheme, we introduce our new KV-SPHF on Cramer-Shoup ciphertexts which immediately leads to a quite efficient instantiation of the Katz and Vaikuntanathan one-round PAKE [24], secure in the BPR model.

2.1 General Definition of SPHFs

Let us consider a language $\mathrm{L} \subseteq \mathcal{S}et$, and some global parameters for the SPHF, assumed to be in the common random string (CRS). The SPHF system for the language L is defined by four algorithms:

- $\mathsf{HashKG}(\mathrm{L})$ generates a hashing key hk for the language L;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C)$ derives the projection key hp, possibly depending on the word C;
- $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)$ outputs the hash value of the word C from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, C, w)$ outputs the hash value of the word C from the projection key hp, and the witness w that $C \in \mathrm{L}$.

The *correctness* of the SPHF assures that if $C \in \mathrm{L}$ with w a witness of this membership, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}, C, w)$. On the other hand, the security is defined through the *smoothness*, which guarantees that, if $C \notin \mathrm{L}$, the hash value is *statistically* indistinguishable from a random element, even knowing hp.

2.2 Smoothness Adaptivity and Key Word-Dependence

This paper will exploit the very strong notion KV-SPHF. Informally, while the GL-SPHF definition allows the projection key hp to depend on the word C, the KV-SPHF definition prevents the projection key hp from depending on C, as in the original CS-SPHF definition. In addition, the smoothness should hold even if C is chosen as an arbitrary function of hp. This models the fact the adversary can see hp before deciding which word C it is interested in. More formal definitions follow, where we denote Π the range of the hash function.

CS-SPHF. This is almost[1] the initial definition of SPHF, where the projection key hp does not depend on the word C (word-independent key), but the word C cannot be chosen after having seen hp for breaking the smoothness (non-adaptive smoothness). More formally, a CS-SPHF is ε-smooth if ProjKG does not use its input C and if, for any $C \in \mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \xleftarrow{\$} \Pi\}.$$

GL-SPHF. This is a relaxation, where the projection key hp can depend on the word C (word-dependent key). More formally, a GL-SPHF is ε-smooth if, for any $C \in \mathcal{S}et \backslash \mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, C)\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, C);\ H \xleftarrow{\$} \Pi\}.$$

[1] In the initial definition, the smoothness was defined for a word C randomly chosen from $\mathcal{S}et \backslash \mathrm{L}$, and not necessarily for any such word.

KV-SPHF. This is the strongest SPHF, in which the projection key hp does not depend on the word C (word-independent key) and the smoothness holds even if C depends on hp (adaptive smoothness). More formally, a KV-SPHF is ε-smooth if ProjKG does not use its input C and, for any function f onto $\mathcal{S}et\backslash\mathrm{L}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathrm{L}, f(\mathsf{hp}))\}$$

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathrm{L});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathrm{L}, \perp);\ H \xleftarrow{\$} \Pi\}.$$

Remark 1. One can see that a perfectly smooth (*i.e.*, 0-smooth) CS-SPHF is also a perfectly smooth KV-SPHF, since each value H has exactly the same probability to appear, and so adaptively choosing C does not increase the above statistical distance. However, as soon as a weak word C can bias the distribution, f can exploit it.

2.3 SPHFs on Languages of Ciphertexts

We could cover languages as general as those proposed in [5], but for the sake of clarity, and since the main applications need some particular cases only, we focus on SPHFs for languages of ciphertexts, whose corresponding plaintexts verify some relations. We denote these languages $\textsc{Lofc}_{\mathsf{full\text{-}aux}}$.

The parameter full-aux will parse in two parts (crs, aux): the public part crs, known in advance, and the private part aux, possibly chosen later. More concretely, crs represents the public values: it will define the encryption scheme (and will thus contain the global parameters and the public key of the encryption scheme) with the global format of both the tuple to be encrypted and the relations it should satisfy, and possibly additional public coefficients; while aux represents the private values (indeed, unless specified differently, as in Section 7, aux is assumed private): it will specify the relations, with more coefficients or constants that will remain private, and thus implicitly known by the sender and the receiver (as the expected password, for example, in PAKE protocols).

To keep aux secret, hp should not leak any information about it. We will thus restrict HashKG and ProjKG not to use the parameter aux, but just crs. This is a stronger restriction than required for our purpose, since one can use aux without leaking any information about it. But we already have quite efficient instantiations, and it makes everything much simpler to present.

2.4 SPHFs on Cramer-Shoup Ciphertexts

Labeled Cramer-Shoup Encryption Scheme (CS). We briefly review the CS labeled encryption scheme, where we combine all the public information in the encryption key. We thus have a group $\mathbb{G}$ of prime order p, with two independent generators $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, a hash function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$ from a collision-resistant hash function family onto $\mathbb{Z}_p^*$, and a reversible mapping $\mathcal{G}$ from $\{0,1\}^n$ to $\mathbb{G}$. From 5 scalars $(x_1, x_2, y_1, y_2, z) \xleftarrow{\$} {\mathbb{Z}_p}^5$, one also sets $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and

$h = g_1^z$. The encryption key is $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$, while the decryption key is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z)$. For a message $m \in \{0,1\}^n$, with $M = \mathcal{G}(m) \in \mathbb{G}$, the labeled Cramer-Shoup ciphertext is:

$$C \stackrel{\text{def}}{=} \mathsf{CS}(\ell, \mathsf{ek}, M; r) \stackrel{\text{def}}{=} (\mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r),$$

with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e) \in \mathbb{Z}_p^*$. If one wants to encrypt a vector of group elements $(M_1, \ldots, M_n)$, all at once in a non-malleable way, one computes all the individual ciphertexts with a common $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$ for $v_1, \ldots, v_n$. Hence, everything done on tuples of ciphertexts will work on ciphertexts of vectors. In addition, the Cramer-Shoup labeled encryption scheme on vectors is IND-CCA under the DDH assumption.

The (known) GL-SPHF for CS. Gennaro and Lindell [20] proposed an SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta, \theta, \mu, \nu) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^4$. The associated projection key, on a ciphertext $C = (\mathbf{u} = (u_1, u_2) = (g_1^r, g_2^r), e = \mathcal{G}(m) \cdot h^r, v = (cd^\xi)^r)$, is $\mathsf{hp} = g_1^\eta g_2^\theta h^\mu (cd^\xi)^\nu \in \mathbb{G}$. Then, one can compute the hash value in two different ways, for the language $\text{L}_{\text{OF}}\text{C}_{\mathsf{ek},m}$ of the valid ciphertexts of $M = \mathcal{G}(m)$, where $\mathsf{crs} = \mathsf{ek}$ is public but $\mathsf{aux} = m$ is kept secret:

$$\begin{aligned} H \stackrel{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\text{def}}{=} u_1^\eta u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu \\ &= \mathsf{hp}^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) \stackrel{\text{def}}{=} H'. \end{aligned}$$

A (new) KV-SPHF for CS. We give here the description of the first known KV-SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^5$; the associated projection key is the pair $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\mu c^\nu, \mathsf{hp}_2 = g_1^{\eta_2} d^\nu) \in \mathbb{G}^2$. Then one can compute the hash value in two different ways, for the language $\text{L}_{\text{OF}}\text{C}_{\mathsf{ek},m}$ of the valid ciphertexts of $M = \mathcal{G}(m)$ under ek:

$$\begin{aligned} H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi\eta_2)} u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu \\ &= (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = H'. \end{aligned}$$

Theorem 2. *The above* SPHF *is a perfectly smooth (*i.e.*, 0-smooth)* KV-SPHF.

The proof can be found in Section 4.1 as an illustration of our new framework.

2.5 An Efficient One-Round PAKE

Review of the Katz and Vaikuntanathan PAKE. As explained earlier, Katz and Vaikuntanathan [24] recently proposed a one-round PAKE scheme. Their general framework follows Gennaro and Lindell [20] approach, which needs an SPHF on a labeled IND-CCA encryption scheme. To allow a SPHF-based PAKE scheme to be one-round, the ciphertext and the SPHF projection key for verifying the correctness of the partner's ciphertext should be sent together, before having seen the partner's ciphertext: the projection key should be independent of the ciphertext. In addition, the adversary can wait until it receives the partner's projection key before generating the ciphertext, and thus a stronger smoothness is required. This is exactly why we need a KV-SPHF in this one-round PAKE framework.

- Players U and U' both use $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$;
- U, with password pw, chooses $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$, computes $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\mu} c^{\nu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\nu})$, sets $\ell = (U, U', \mathsf{hp})$, and generates $C = (\mathbf{u} = (g_1^r, g_2^r), e = \mathcal{G}(\mathsf{pw}) \cdot h^r, v = (cd^{\xi})^r)$ with r a random scalar in $\mathbb{Z}_p$ and $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$. U sends $\mathsf{hp} \in \mathbb{G}^2$ and $C \in \mathbb{G}^4$ to U';
- Upon receiving $\mathsf{hp}' = (\mathsf{hp}'_1, \mathsf{hp}'_2) \in \mathbb{G}^2$ and $C' = (\mathbf{u}' = (u'_1, u'_2), e', v') \in \mathbb{G}^4$ from U', U sets $\ell' = (U', U, \mathsf{hp}')$ and $\xi' = \mathfrak{H}_K(\ell', \mathbf{u}', e')$ and computes
$$\mathsf{sk}_U = {u'_1}^{(\eta_1 + \xi' \eta_2)} {u'_2}^{\theta} (e'/\mathcal{G}(\mathsf{pw}))^{\mu} {v'}^{\nu} \cdot (\mathsf{hp}'_1 {\mathsf{hp}'_2}^{\xi})^r.$$

Fig. 1. One-Round PAKE based on DDH

Our Construction. Our KV-SPHF on Cramer-Shoup ciphertexts can be used in the Katz and Vaikuntanathan framework for PAKE [24]. It leads to the most efficient PAKE known so far, and it is *one-round*. Each user indeed only sends 6 elements of $\mathbb{G}$ (see Figure 1), instead of 70 elements of $\mathbb{G}$ for the Katz and Vaikuntanathan's instantiation using a Groth-Sahai SS-NIZK [21], or 20 group elements for the Jutla and Roy's [22] improvement using a relatively-sound NIZK.

The formal security result follows from the Theorem 4 in Section 6. We want to insist that our construction does not need pairing-friendly groups, and the plain DDH assumption is enough, whereas the recent constructions made heavy use of pairing-based proofs *à la* Groth-Sahai.

3 Computational Smoothness and Definition of TSPHF

In order to build a one-round PAKE provably secure in the UC framework, one needs the simulator to be able to compute the hash value even when a dummy password has initially been committed, whereas a success is expected. Katz and Vaikuntanathan [24] thus asked the players to add an encryption of the hashing key together with the projection key, and a proof a correctness (a simulation-sound extractable NIZK proof). We now improve on this technique.

More precisely, in this section, we introduce TSPHFs, which are SPHFs with a trapdoor enabling a simulator to compute the hash value on any word C without knowing hk nor any witness, but only knowing hp. TSPHFs also provide a way to ensure that hp is valid. It can be seen that, intuitively, in most cases, a TSPHF cannot be statistically smooth, and so, before introducing TSPHFs, we need to introduce a new notion of smoothness: computational smoothness.

3.1 Computationally-Smooth SPHF

Let us first suppose there exists an algorithm Setup which takes as input the security parameter $\mathfrak{K}$ and outputs a CRS crs together with a trapdoor τ, which is not the trapdoor of the TSPHF, but just a trapdoor of crs. The trapdoor τ can be $\perp$, but in our article, the trapdoor will contain the decryption key of the

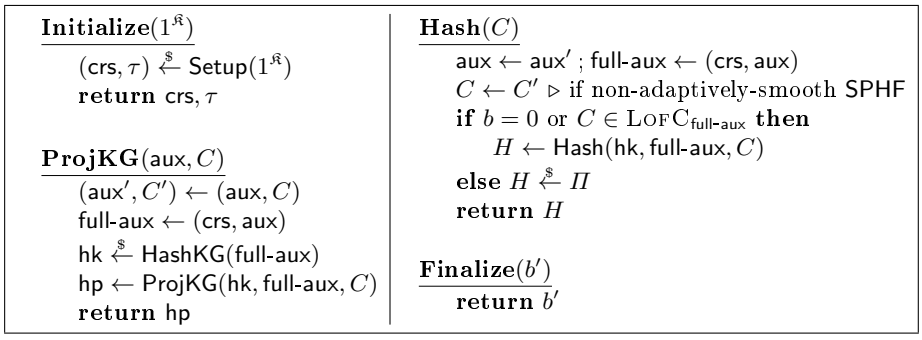

Fig. 2. Games $\mathsf{Exp}^{\text{smooth}-b}_{\mathfrak{K}}(\mathcal{A})$ ($b = 0$ or 1) for computational smoothness

encryption scheme, and possibly other data such that, for any $C \in \mathcal{S}et$, it is possible to check whether $C \in \text{Lof C}_{\mathsf{full\text{-}aux}}$ or not, in polynomial time.

Let us then consider the two games $\mathsf{Exp}^{\text{smooth}-b}_{\mathfrak{K}}(\mathcal{A})$ (with $b = 0$ or 1) depicted in Figure 2, where Π denotes the set of hash values. There are two variants of the games: whether the SPHF is adaptively-smooth (KV-SPHF) or not (CS-SPHF and GL-SPHF).

Let us first explain the games for a non-adaptively-smooth SPHF. The procedure **Initialize** generates and outputs the CRS crs and its trapdoor τ. It is important to notice that computational smoothness has to hold even when the adversary knows the trapdoor, and so may depend on what is in the trapdoor τ.

During the execution of the game, the adversary is allowed to make one query **ProjKG**(aux, C) to get a projection key hp associated with aux and C, and then one query **Hash**$(\perp)$ to get the hash value of C. If $C \in \text{Lof C}_{\mathsf{full\text{-}aux}}$, smoothness does not apply, thus **Hash**(C) really returns the hash value H of C: $H = \mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C)$, for a hashing key hk associated with hp. Otherwise, the smoothness should apply with a real-or-random indistinguishability game, and thus, if $b = 0$ the real hash value is returned too, whereas a random value in Π is returned when $b = 1$. Eventually, the adversary ends the game by querying the **Finalize** procedure with its guess b' for b. We remark that the procedure **Hash** may or may not be polynomial time, depending on τ, since it is not necessarily possible to efficiently check whether $C \in \text{Lof C}_{\mathsf{full\text{-}aux}}$.

For the adaptively-smooth variant, the adversary does not need to provide the word C when it makes a query to **ProjKG**. It gives $\perp$ instead and can choose C adaptively after having seen hp, as input to the **Hash** query.

Formally, an SPHF is (t, ε)-smooth if for all adversary $\mathcal{A}$ running in time at most t:

$$\left|\Pr\left[\mathsf{Exp}^{\text{smooth}-1}_{\mathfrak{K}}(\mathcal{A}) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{smooth}-0}_{\mathfrak{K}}(\mathcal{A}) = 1\right]\right| \leq \varepsilon.$$

The classical statistical-smoothness implies the (t, ε)-smoothness for any t, and any non-negligible ε (and whatever is the trapdoor τ).

3.2 Trapdoor SPHF

A TSPHF is an extension of a classical SPHF with an additional algorithm TSetup, which takes as input the CRS crs and outputs an additional CRS crs′ and a trapdoor τ' specific to crs′, which can be used to compute the hash value of words C knowing only hp. For TSPHF, we assume full-aux = (crs, crs′, aux), although the language $\text{LoF}\text{C}_{\mathsf{full\text{-}aux}}$ still does not depend on crs′. Formally, a TSPHF is defined by seven algorithms:

- TSetup(crs) takes as input the CRS crs (generated by Setup) and generates the second CRS crs′, together with a trapdoor τ';
- HashKG, ProjKG, Hash, and ProjHash behave as for a classical SPHF;
- VerHP(hp, full-aux, C) outputs 1 if hp is a valid projection key, and 0 otherwise. When hp does not depend on C (word-independent key), the input C can be replaced by $\perp$;
- THash(hp, full-aux, C, τ') outputs the hash value of C from the projection key hp and the trapdoor τ'.

It must verify the following properties:

- *Correctness* is defined by two properties: *hash correctness*, which corresponds to correctness for classical SPHFs, and an additional property called *trapdoor correctness*, which states that, for any $C \in \mathcal{S}et$, if hk and hp are honestly generated, we have: VerHP(hp, full-aux, C) = 1 and Hash(hk, full-aux, C) = THash(hp, full-aux, C, τ'), with overwhelming probability;
- *Smoothness* is exactly the same as for SPHFs, except that in the **Initialize** procedure, TSetup is also called, but while τ' is dropped, crs′ is forwarded to the adversary (together with crs and τ);
- The (t, ε)-*soundness* property says that, given crs, τ and crs′, no adversary running in time at most t can produce a projection key hp, a value aux, a word C and valid witness w such that VerHP(hp, full-aux, C) = 1 but THash(hp, full-aux, C, τ') $\neq$ ProjHash(hp, full-aux, C, w), with probability at least ε. The perfect soundness states that the property holds for any t and any $\varepsilon > 0$.

It is important to notice that τ is not an input of THash and it is possible to use THash, while generating crs with an algorithm which cannot output τ (as soon as the distribution of crs output by this algorithm is indistinguishable from the one output by Setup, obviously). For example, if τ contains a decryption key, it is still possible to use the IND-CPA game for the encryption scheme, while making calls to THash.

3.3 A Naive Construction of TSPHFs using NIZK

A naive solution to transform any SPHF into a TSPHF consists in replacing the projection key hp by a pair (hp, π), where π is an extractable NIZK (ENIZK) proof of the knowledge of a hashing key hk such that hp is the projection key

of hk. This is essentially the approach of [24]. In the TSPHF full version [7], we show that this provides a correct, smooth and sound TSPHF. Intuitively the hash correctness directly comes from the correctness of the original SPHF, the trapdoor correctness and the soundness come from the extractability of the ENIZK proof (and may not be perfect) and the smoothness comes from the zero-knowledge property of the ENIZK proof. We also show some improvements for this naive construction to make quite efficient TSPHFs, and in particular to avoid having to do bit-by-bit Groth-Sahai ENIZK proofs. These improvements can be seen as a generalization of the method proposed by Jutla and Roy in [22, Section 8]. But even with these improvements, this naive construction is still less efficient than the constructions described in the sequel.

4 Construction of DDH-based TSPHFs

In the SPHF full version [6], we propose a formal framework for SPHFs using a new notion of graded rings, derived from [19]. It enables to deal with cyclic groups, bilinear groups (with symmetric or asymmetric pairings), or even groups with multi-linear maps. In particular, it helps to construct concrete SPHFs for quadratic pairing equations over ciphertexts, which enable to construct efficient LAKE [5] for any language handled by the Groth-Sahai NIZK proofs, and so for any NP-language (see Section 6.1).

However, we focus here on cyclic groups, with the basic intuition only, and provide some illustrations. While we keep the usual multiplicative notation for the cyclic group $\mathbb{G}$, we use an extended notation: $r \odot u = u \odot r = u^r$, for $r \in \mathbb{Z}_p$ and $u \in \mathbb{G}$, and $u \oplus v = u \cdot v$, for $u, v \in \mathbb{G}$. Basically, $\oplus$ and $\odot$ correspond to the addition and the multiplication in the exponents, that are thus both commutative. We then extend this notation in a natural way when working on vectors and matrices.

4.1 Generic Framework for GL-SPHF/KV-SPHF

Our goal is to deal with languages of ciphertexts $\text{LofC}_{\mathsf{full\text{-}aux}}$: we assume that crs is fixed and we write $\mathrm{L}_{\mathsf{aux}} = \text{LofC}_{\mathsf{full\text{-}aux}} \subseteq \mathcal{S}et$ where $\mathsf{full\text{-}aux} = (\mathsf{crs}, \mathsf{aux})$.

Language Representation. For a language $\mathrm{L}_{\mathsf{aux}}$, we assume there exist two positive integers k and n, a function $\Gamma : \mathcal{S}et \mapsto \mathbb{G}^{k \times n}$, and a family of functions $\Theta_{\mathsf{aux}} : \mathcal{S}et \mapsto \mathbb{G}^{1 \times n}$, such that for any word $C \in \mathcal{S}et$, $(C \in \mathrm{L}_{\mathsf{aux}}) \iff (\exists \boldsymbol{\lambda} \in \mathbb{Z}_p^{1 \times k}$ such that $\Theta_{\mathsf{aux}}(C) = \boldsymbol{\lambda} \odot \Gamma(C))$. In other words, we assume that $C \in \mathrm{L}_{\mathsf{aux}}$, if and only if, $\Theta_{\mathsf{aux}}(C)$ is a linear combination of (the exponents in) the rows of some matrix $\Gamma(C)$. For a KV-SPHF, Γ is supposed to be a constant function (independent of the word C). Otherwise, one gets a GL-SPHF.

We furthermore require that a user, who knows a witness w of the membership $C \in \mathrm{L}_{\mathsf{aux}}$, can efficiently compute the above *linear* combination $\boldsymbol{\lambda}$. This may seem a quite strong requirement but this is actually verified by very expressive languages over ciphertexts such as ElGamal, Cramer-Shoup and variants.

We briefly illustrate it on our KV-SPHF on CS: $C = (u_1 = g_1^r, u_2 = g_2^r, e = M \cdot h^r, v = (cd^\xi)^r)$, with $k = 2$, $\mathsf{aux} = M$ and $n = 5$:

$$\Gamma = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \qquad \boldsymbol{\lambda} = (r, r\xi) \qquad \begin{array}{r} \boldsymbol{\lambda} \odot \Gamma = (g_1^r, g_1^{r\xi}, g_2^r, h^r, (cd^\xi)^r) \\ \Theta_M(C) = (u_1, u_1^\xi, u_2, e/M, v). \end{array}$$

Essentially, one tries to make the first columns of $\Gamma(C)$ and the first components of $\Theta_{\mathsf{aux}}(C)$ to completely determine $\boldsymbol{\lambda}$. In our illustration, the first two columns with $u_1 = g_1^r$ and $u_1^\xi = g_1^{r\xi}$ really imply $\boldsymbol{\lambda} = (r, r\xi)$, and the three last columns help to check the language membership: we want $u_2 = g_2^r$, $e/M = h^r$, and $v = (cd^\xi)^r$, with the same r as for u_1.

Smooth Projective Hash Function. With the above notations, the hashing key is a vector $\mathsf{hk} = \boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)^\intercal \xleftarrow{\$} \mathbb{Z}_p^n$, while the projection key is, for a word C, $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \odot \boldsymbol{\alpha} \in \mathbb{G}^k$ (if Γ depends on C, this leads to a GL-SPHF, otherwise, one gets a KV-SPHF). Then, the hash value is:

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \stackrel{\text{def}}{=} \Theta_{\mathsf{aux}}(C) \odot \boldsymbol{\alpha} = \boldsymbol{\lambda} \odot \boldsymbol{\gamma}(C) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w).$$

Our above Γ, $\boldsymbol{\lambda}$, and Θ_M immediately lead to our KV-SPHF on CS from the Section 2.4: with $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$, the product with Γ leads to: $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\mu c^\nu, \mathsf{hp}_2 = g_1^{\eta_2} d^\nu) \in \mathbb{G}^2$, and

$$\begin{aligned} H = \mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) &\stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi\eta_2)} u_2^\theta (e/\mathcal{G}(m))^\mu v^\nu \\ &= (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = H'. \end{aligned}$$

The generic framework detailed in the SPHF full version [6] also contains a security analysis that proves the above generic SPHF is perfectly smooth: Intuitively, for a word $C \notin \mathrm{L}_{\mathsf{aux}}$ and a projection key $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \odot \boldsymbol{\alpha}$, the vector $\Theta_{\mathsf{aux}}(C)$ is not in the linear span of $\Gamma(C)$, and thus $H = \Theta_{\mathsf{aux}}(C) \odot \boldsymbol{\alpha}$ is independent from $\Gamma(C) \odot \boldsymbol{\alpha} = \mathsf{hp}$. This also proves the Theorem 2 as a particular case.

4.2 Efficient Construction of TSPHFs under DDH

We now explain how to construct a TSPHF in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, from any SPHF constructed via the above framework, provided the SPHF does not require pairings (as all the SPHFs described in this paper), and under an additional assumption detailed later (for the smoothness to hold). To this aim, we extend our notations with $g_1 \odot g_2 = g_2 \odot g_1 = e(g_1, g_2)$, and scalars can operate on any group element as before. Intuitively, our TSPHF construction is such that all the "SPHF" part of the TSPHF is in $\mathbb{G}_1$, whereas the trapdoor part is in $\mathbb{G}_2$. And the trapdoor part simply contains some representation of $\boldsymbol{\alpha}$, representation which cannot be used without knowing the trapdoor τ'.

The second CRS is a random element $\mathsf{crs}' = \zeta \xleftarrow{\$} \mathbb{G}_2$, and its trapdoor is its discrete logarithm τ', such that $\zeta = g_2^{\tau'} = \tau' \odot g_2$. The hashing key $\mathsf{hk} = \boldsymbol{\alpha}$ is

the same as before. The projection key is the ordered pair $\mathsf{hp} = (\boldsymbol{\gamma}, \boldsymbol{\chi})$, where $\boldsymbol{\gamma}$ is the same as before, and $\boldsymbol{\chi} = \zeta \odot \boldsymbol{\alpha}$. The projection key is valid (*i.e.*, $\mathsf{VerHP}(\mathsf{hp}, \mathsf{full\text{-}aux}, C) = 1$) if and only if

$$\boldsymbol{\chi} \in \mathbb{G}_2^n \qquad \text{and} \qquad \zeta \odot \boldsymbol{\gamma} = \Gamma \odot \boldsymbol{\chi}, \tag{1}$$

Then, for any word $C \in \mathrm{L}_{\mathsf{full\text{-}aux}}$ with witness w corresponding to the vector $\boldsymbol{\lambda}$, the hash value is

$$\mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C) \stackrel{\text{def}}{=} \Theta(C) \odot \boldsymbol{\alpha} \odot g_2 = \boldsymbol{\lambda} \odot \boldsymbol{\gamma} \odot g_2 \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, w).$$

Equation (1) means that $\boldsymbol{\chi}$ can be written $\boldsymbol{\chi} = \tau' \odot \boldsymbol{\alpha'}$, with $\boldsymbol{\alpha'} \in \mathbb{Z}_p^n$ verifying $\boldsymbol{\gamma} = \Gamma \odot \boldsymbol{\alpha'}$, *i.e.*, $\mathsf{hk}' = \boldsymbol{\alpha'}$ is a valid hashing key for $\boldsymbol{\gamma}$. We do not have necessarily $\boldsymbol{\alpha} = \boldsymbol{\alpha'}$, however, for any word $C \in \mathrm{L}_{\mathsf{full\text{-}aux}}$, we have and we set

$$\boldsymbol{\lambda} \odot \boldsymbol{\gamma} \odot g_2 = \Theta(C) \odot \boldsymbol{\alpha'} \odot g_2 = \tau'^{-1} \odot \Theta(C) \odot \boldsymbol{\chi} \stackrel{\text{def}}{=} \mathsf{THash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, \tau').$$

In the TSPHF full version [7], we prove the resulting TSPHF is computationally smooth under the DDH assumption in $\mathbb{G}_2$, if the discrete logarithms of $\Gamma_{\mathsf{aux}}(C)$ can be computed from τ. This latter assumption on $\Gamma_{\mathsf{aux}}(C)$ and τ is required for technical reasons in the proof of smoothness. The correctness and the perfect soundness are easy to prove from the construction, and so the resulting TSPHF is correct, smooth and sound.

4.3 TSPHF on Cramer-Shoup Ciphertexts

We apply this technique to extend the SPHF on Cramer-Shoup ciphertexts from Section 2 into a TSPHF. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a bilinear group. We consider the same language and use the same notations as in Section 2 except we replace $\mathbb{G}$ by $\mathbb{G}_1$, g_1 and g_2 by $g_{1,1}$ and $g_{1,2}$ resp., and h by h_1, while g_2 is a generator of $\mathbb{G}_2$.

To get a TSPHF, we choose a random scalar τ' in $\mathbb{Z}_p$ and set $\mathsf{crs}' = \zeta = g_2^{\tau'}$. Then the hashing key, the projection key and the hash value of the TSPHF are defined as follows:

$$\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$$

$$\mathsf{hp} = (\mathsf{hp}_1 = g_{1,1}^{\eta_1} g_{1,2}^{\theta} h_1^{\mu} c^{\nu}, \mathsf{hp}_2 = g_{1,1}^{\eta_2} d^{\nu}, \mathsf{hp}_3) \in \mathbb{G}_1^2 \times \mathbb{G}_2^5$$

$$\text{where } \mathsf{hp}_3 = (\chi_{1,1} = \zeta^{\eta_1}, \chi_{1,2} = \zeta^{\eta_2}, \chi_2 = \zeta^{\theta}, \chi_3 = \zeta^{\mu}, \chi_4 = \zeta^{\nu}) \in \mathbb{G}_2^5$$

$$\mathsf{Hash}(\mathsf{hk}, (\mathsf{ek}, m), C) = e({u_1}^{(\eta_1 + \xi\eta_2)} {u_2}^{\theta} (e/\mathcal{G}(m))^{\mu} v^{\nu}, g_2)$$

$$\mathsf{ProjHash}(\mathsf{hp}, (\mathsf{ek}, m), C, r) = e((\mathsf{hp}_1 {\mathsf{hp}_2}^{\xi})^r, g_2)$$

The projection key is valid if and only if: $e(\mathsf{hp}_1, \zeta) = e(g_{1,1}, \chi_{1,1}) \cdot e(g_{1,2}, \chi_2) \cdot e(h_1, \chi_3) \cdot e(c, \chi_4)$ and $e(\mathsf{hp}_2, \zeta) = e(g_{1,1}, \chi_{1,2}) \cdot e(d, \chi_4)$. For any $C \in \mathrm{LOFC}_{(\mathsf{crs},\mathsf{aux})}$, the hash value can be computed from C and τ' as $\mathsf{THash}(\mathsf{hp}, (\mathsf{ek}, m), C, \tau')$:

$$\Big(e(u_1, \chi_{1,1} \cdot \chi_{1,2}^{\xi}) \cdot e(u_2, \chi_2) \cdot e(e/\mathcal{G}(m), \chi_3) \cdot e(v, \chi_4)\Big)^{1/\tau'}.$$

The resulting TSPHF is smooth under the DDH in $\mathbb{G}_2$, hence the global SXDH assumption. More complex and concrete examples of TSPHFs can be found in the TSPHF full version [7].

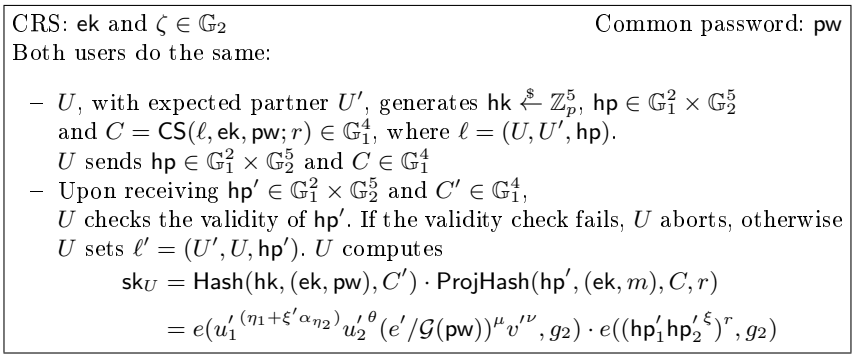

Fig. 3. UC-Secure One-Round PAKE based on DDH

4.4 One-Round UC-Secure PAKE from TSPHF

We now show how our TSPHF can lead to a very efficient one-round PAKE, secure in the UC framework with static corruptions. This is a slight variant of the one-round PAKE from [24], where the SPHF and the SS-NIZK proofs are replaced by a TSPHF, which can be much more efficient; and where, as in our previous PAKE, we use the Cramer-Shoup encryption scheme as commitment scheme, instead of the original inefficient IND-CCA encryption scheme based on the Naor-Yung principle [26]. The concrete scheme is depicted in Figure 3. The communication complexity is of 6 elements in $\mathbb{G}_1$ and 5 elements in $\mathbb{G}_2$ only in each direction.

We show in the TSPHF full version [7] that the protocol actually works with any TSPHF on an IND-CCA encryption scheme, and provide a full generic proof. It is in the same vein as the KV's proof but a bit more intricate for two reasons: we do not assume a prior agreement of the session ID which makes our scheme a truly one-round protocol; our TSPHF does not guarantee the smoothness (even computationally) when the trapdoor τ' is known, and then, we have to modify the order of the games to use this trapdoor at the very end only.

One can remark that the original scheme in [24] can be seen as an instantiation of our scheme with the naive TSPHF based on NIZK (Section 3.3). Therefore, the security of the original KV's PAKE protocol is actually implied by our proof. And our proof also shows that their construction can be simplified by removing the commitment of hk and replacing the SS-NIZK by an ENIZK proof of knowledge of hk.

5 More Constructions of SPHFs

In this section, we first illustrate more our generic framework, by constructing more evolved SPHFs, and then we show some interesting applications. One can note that all these constructions are without pairings, the generic framework can thus be used to extend them with trapdoors, with some more applications presented in Section 7.

5.1 KV-SPHF for Linear Multi-Exponentiation Equations

We present several instantiations of KV-SPHFs, in order to illustrate our framework, but also to show that our one-round PAKE protocol from Section 2.5 can be extended to one-round LAKE [5]. In PAKE/LAKE, we use SPHFs to prove that the plaintexts associated with some ElGamal-like ciphertexts verify some relations. The communication complexity of these protocols depends on the ciphertexts size and of the projection keys size. We first focus on ElGamal ciphertexts, and then explain how to handle Cramer-Shoup ciphertexts. More constructions are detailed in the SPHF full version [6].

Notations. We work in a group $\mathbb{G}$ of prime order p, generated by g, in which we assume the DDH assumption to hold. We define ElGamal encryption scheme with encryption key $\mathsf{ek} = (g, h = g^x)$. Let n, m and t be three positive integers. In the following i, j and k always range from 1 to n, from 1 to m and from 1 to t respectively in all the products $\prod_i$, $\prod_j$, $\prod_k$ and tuples $(\cdot)_i$, $(\cdot)_j$, $(\cdot)_k$. We are interested in languages of the ciphertexts $C_i = (u_i = g^{r_i}, e_i = h^{r_i} \cdot X_i)$, for which $X_1, \ldots, X_n \in \mathbb{G}$ satisfy

$$\begin{array}{r} \exists y_1, \ldots, y_m \in \mathbb{Z}_p, \\ \forall k \in \{1, \ldots, t\}, \end{array} \prod_{i=1}^{n} X_i^{a_{k,i}} \cdot \prod_{j=1}^{m} A_{k,j}^{y_j} = B_k, \text{ with } \begin{array}{r} \mathsf{crs} = (p, \mathbb{G}, \mathsf{ek}, (A_{k,j})_{k,j}) \\ \mathsf{aux} = ((a_{k,i})_{k,i}, (B_k)_k), \end{array} \quad (2)$$

where $(A_{k,j})_{k,j} \in \mathbb{G}^{t \times m}$ are public and known in advance (*i.e.*, are in crs), while $((a_{k,i})_{k,i}, (B_k)_k) \in \mathbb{Z}_p^{t \times n} \times \mathbb{G}^t$ can be kept secret (*i.e.*, can be in aux). This can be seen as a system of t linear multi-exponentiation equations.

The Groth-Sahai Approach. Naive use of the Groth Sahai framework invites us to also commit to scalars as $Y_j = g^{y_j}$ and to show that the plaintexts $(X_i)_i$ and $(Y_j)_j$ satisfy:

$$\begin{array}{c} \exists y_1, \ldots, y_m \in \mathbb{Z}_p, \forall k \in \{1, \ldots, t\}, \prod_i X_i^{a_{k,i}} \cdot \prod_j A_{k,j}^{y_j} = B_k, \\ \forall j \in \{1, \ldots, m\}, Y_j = g^{y_j}. \end{array}$$

Since there is no efficient way to extract y_j from Y_j, committing to y_j is often not useful.

A First SPHF. We thus consider the language of the ciphertexts $C_i = (u_i = g^{r_i}, e_i = h^{r_i} \cdot X_i)$, for $X_1, \ldots, X_n \in \mathbb{G}$ satisfying (2). The witnesses are $(X_i)_i$, $(r_i)_i$ and $(y_j)_j$, or just $(r_i)_i$ and $(y_j)_j$. The matrix Γ is the following one:

$$\Gamma = \left(\begin{array}{cccc|cccc} g & & & 1 & h & & & 1 \\ & \ddots & & & & \ddots & & \\ & & g & & & & h & \\ 1 & & & & 1 & & & \\ \hline & & & & A_{1,1}^{-1} & \cdots & & A_{t,1}^{-1} \\ & & 1 & & \vdots & & & \vdots \\ & & & & A_{1,m}^{-1} & \cdots & & A_{t,m}^{-1} \end{array}\right)$$

$$\Theta_{\mathsf{aux}}(\boldsymbol{C}) = \left(\left(\prod_i u_i^{a_{k,i}}\right)_k, \left(\prod_i e_i^{a_{k,i}} / B_k\right)_k\right)$$

$$\boldsymbol{\lambda} = \left(\left(\sum_i a_{k,i} r_i\right)_k, (y_j)_j\right)$$

$$\boldsymbol{\lambda} \odot \Gamma = \left(\left(\prod_i g^{a_{k,i} r_i}\right)_k, \left(\prod_i h^{a_{k,i} r_i} / \prod_j A_{k,j}^{y_j}\right)_k\right)$$

The upper-left diagonal block imposes the first t values on $\boldsymbol{\lambda}$, while the last t columns define the t relations: The last t components of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$, namely $\prod_i e_i^{a_{k,i}}/B_k = \prod_i h^{a_{k,i}r_i} \cdot \prod_i X_i^{a_{k,i}}/B_k$ (for $k = 1, \ldots, t$), are equal to the last t components of $\boldsymbol{\lambda} \odot \Gamma$, namely $\prod_i h^{a_{k,i}r_i}/\prod_j A_{k,j}^{y_j}$ ($(y_j)_j$ are just the last t components of $\boldsymbol{\lambda}$), if and only if the relations in (2) are all satisfied. It thus leads to the following KV-SPHF, with $(\mathsf{hp}_{1,k} = g^{\eta_k} h^{\mu_k})_k$ and $(\mathsf{hp}_{2,j} = \prod_k A_{k,j}^{-\mu_k})_j$, for $\mathsf{hk} = ((\eta_k)_k, (\mu_k)_k)$:

$$H = \prod_k \left(\prod_i (u_i^{\eta_k} e_i^{\mu_k})/B_k^{\mu_k} \right) = \prod_k \mathsf{hp}_{1,k}^{\sum_i a_{k,i} r_i} \cdot \prod_j \mathsf{hp}_{2,j}^{y_j} = H'.$$

As a consequence, the ciphertexts and the projection keys globally consist of $2n+t+m$ elements from $\mathbb{G}$ only. This is much more compact than the $2n+4m+t$ elements one would get by additionally committing the $(Y_j = g^{y_j})_j$.

Ciphertexts with Randomness Reuse. In some cases, even the constants $(a_{k,i})_{k,i}$ can be public and known in advance, and thus moved from aux to crs. In this case, one can furthermore shorten ElGamal ciphertexts by using multiple independent encryption keys for encrypting the X_i's: $\mathsf{ek}_i = (g, h_i = g^{x_i})$, for $i = 1, \ldots, m$. This allows to reuse the same random coins [2]. More precisely, we are now interested in the language of the ciphertexts $C = (u = g^r, (e_i = h_i^r \cdot X_i)_i)$, for $X_1, \ldots, X_n \in \mathbb{G}$ still satisfying (2). This improves on the length of the ciphertexts, from $2n$ group elements in $\mathbb{G}$ to $n+1$, and the t first rows of the matrix can be combined into the unique row $(g, \prod_i h_i^{a_{1,i}}, \ldots, \prod_i h_i^{a_{t,i}})$. It thus leads to the following KV-SPHF, with $\mathsf{hp}_1 = g^{\eta} \cdot \prod_k (\prod_i h_i^{a_{k,i}})^{\mu_k}$, $(\mathsf{hp}_{2,j} = \prod_k A_{k,j}^{-\mu_k})_j$, for $\mathsf{hk} = (\eta, (\mu_k)_k)$:

$$H = u^{\eta} \cdot \prod_k \left(\prod_i e_i^{a_{k,i}}/B_k \right)^{\mu_k} = \mathsf{hp}_1^r \cdot \prod_j \mathsf{hp}_{2,j}^{y_j} = H'.$$

Globally, the ciphertexts and the projection keys consist of $n+m+2$ elements from $\mathbb{G}$: this is independent of the number of equations. This randomness-reuse technique will be exploited in Section 6.2 for improving blind signature schemes.

From ElGamal to Cramer-Shoup Encryption. In order to move from ElGamal ciphertexts to Cramer-Shoup ciphertexts (when non-malleability is required), if one already has Γ, Θ_{aux}, and $\boldsymbol{\Lambda}$ to guarantee that the ElGamal plaintexts satisfy a relation, one simply has to make a bigger matrix, diagonal per blocks, with the block Γ and the smaller blocks $(\Gamma_k)_k$ for every ciphertext C_k, where each block Γ_k is the Cramer-Shoup matrix from Section 4.1 without the fourth column (the column with h). The initial matrix Γ guarantees the relations on the ElGamal sub-ciphertexts, and the matrices Γ_k guarantee the validity of the Cramer-Shoup ciphertexts. Since some witnesses are the same, some rows/columns can be packed together. More complex languages on Cramer-Shoup ciphertexts will be exploited in Section 6.1, we thus illustrate how the above combination can be optimized in the case of multi-exponentiation equations.

KV-SPHF on Cramer-Shoup Ciphertexts for Linear Multi-Exponentiation Equations. Let us convert the above KV-SPHF to Cramer-Shoup ciphertexts, with some optimizations: we write $C = (u_1 = g_1^r, u_2 = {g_2}^r, e_1 = h_1^r \cdot X_1, \ldots, e_n = h_n^r \cdot X_n, v = (cd^\xi)^r)$, where $\xi = \mathfrak{H}_K(u_1, u_2, e_1, \ldots, e_n) \in \mathbb{Z}_p^*$, for $X_1, \ldots, X_n \in \mathbb{G}$ satisfying (2). We make the matrix more compact as follows, with $\boldsymbol{\lambda} = (r, r\xi, (y_j)_j)$:

$$\Gamma = \left(\begin{array}{c|c|c|c|c} g_1 & 1 & g_2 & \prod_i h_i^{a_{1,i}} \ldots \prod_i h_i^{a_{t,i}} & c \\ \hline 1 & g_1 & 1 & 1 \ \ldots \ 1 & d \\ \hline 1 & 1 & 1 & A_{1,1}^{-1} \cdots A_{t,1}^{-1} & 1 \\ \vdots & \vdots & \vdots & \vdots \quad \vdots & \vdots \\ 1 & 1 & 1 & A_{1,m}^{-1} \cdots A_{t,m}^{-1} & 1 \end{array}\right).$$

This leads to the following KV-SPHF, with $\mathsf{hp}_1 = g_1^{\eta_1} {g_2}^\theta \cdot \prod_k (\prod_i h_i^{a_{k,i}})^{\mu_k} \cdot c^\nu$, $\mathsf{hp}_2 = g_1^{\eta_2} d^\nu$, $(\mathsf{hp}_{3,j} = \prod_k A_{k,j}^{-\mu_k})_j$, for $\mathsf{hk} = (\eta_1, \eta_2, \theta, (\mu_k)_k, \nu)$:

$$H = u_1^{\eta_1 + \xi\eta_2} \cdot u_2^\theta \cdot \prod\nolimits_k \big(\prod\nolimits_i e_i^{a_{k,i}} / B_k\big)^{\mu_k} \cdot v^\nu = (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \cdot \prod\nolimits_j \mathsf{hp}_{3,j}^{y_j} = H'.$$

5.2 GL-SPHF on Bit Encryption

Our general framework allows to construct KV-SPHFs for any language handled by the Groth-Sahai NIZK proofs (see the SPHF full version [6]). While these KV-SPHFs encompass the language of ciphertexts encrypting a bit, they require pairing evaluations. We show here a more efficient GL-SPHF for bit encryption, which does not need pairings.

Let us consider an ElGamal ciphertext $C = (u = g^r, e = h^r g^y)$, in which one wants to prove that $y \in \{0, 1\}$. We can define the following matrix that depends on C, hence a GL-SPHF:

$$\Gamma(C) = \begin{pmatrix} g & h & 1 & 1 \\ 1 & g & u & e/g \\ 1 & 1 & g & h \end{pmatrix} \qquad \begin{array}{r} \Theta_{\mathsf{aux}}(C) = (u, e, 1, 1) \qquad \boldsymbol{\lambda} = (r, y, -ry) \\ \boldsymbol{\lambda} \odot \Gamma(C) = (g^r, h^r g^y, (u/g^r)^y, (e/gh^r)^y) \end{array}$$

Because of the triangular block in $\Gamma(C)$, one sees that $\Theta_{\mathsf{aux}}(C) = \boldsymbol{\lambda} \odot \Gamma(C)$ if and only if $g^{y(y-1)} = 1$, and thus that $y \in \{0, 1\}$. With $\mathsf{hp}_1 = g^\nu h^\theta$, $\mathsf{hp}_2 = g^\theta u^\eta (e/g)^\lambda$, and $\mathsf{hp}_3 = g^\eta h^\lambda$, for $\mathsf{hk} = (\nu, \theta, \eta, \lambda)$: $H = u^\nu e^\theta = \mathsf{hp}_1^r \cdot \mathsf{hp}_2^y / \mathsf{hp}_3^{ry} = H'$.

6 More Applications of SPHFs

6.1 One-Round LAKE

Since we have shown that our framework allows to design KV-SPHFs for complex languages, we extend our PAKE protocol to LAKE [5]. To this aim, we provide a new security model, inspired from BPR [3] and a complete security proof, which implies the security of our PAKE protocol from Section 2.5.

Review of Language-Authenticated Key Exchange. LAKE is a general framework [5] that generalizes AKE primitives: each player U owns a word W in a certain language $\mathcal{L}$ and expects the other player to own a word W' in a language $\mathcal{L}'$. If everything is compatible (*i.e.*, the languages are the expected languages and the words are indeed in the appropriate languages), the players compute a common high-entropy secret key, otherwise they learn nothing about the partner's values. In any case, external eavesdroppers do not learn anything, even not the outcome of the protocol: did it succeed or not?

More precisely, we assume the two players have initially agreed on a common public part pub for the languages, but then they secretly parametrize the languages with the private parts priv: $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$ is the language they want to use, and $\mathcal{L}_{\mathsf{pub},priv'}$ is the language they assume the other player will use. In addition, each player owns a word W in his language. We will thus have to use SPHFs on ciphertexts on W, priv and $priv'$, with a common $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$ and aux with the private parameters. For simple languages, this encompasses PAKE and Verifier-based PAKE. We refer to [5] for more applications of LAKE.

A New Security Model for LAKE. The first security model for LAKE [5] has been given in the UC framework [13], as an extension of the UC security for PAKE [14]. In this paper, we propose an extension of the PAKE security model presented by Bellare, Pointcheval, and Rogaway [3] model for LAKE: the adversary $\mathcal{A}$ plays a find-then-guess game against n players $(P_i)_{i=1,\ldots,n}$. It has access to several instances Π_U^s for each player $U \in \{P_i\}$ and can activate them (in order to model concurrent executions) via several queries: Execute-queries model passive eavesdroppings; Send-queries model active attacks; Reveal-queries model a possible bad later use of the session key; the Test-query models the secrecy of the session key. The latter query has to be asked to a *fresh* instance (which basically means that the session key is not trivially known to the adversary) and models the fact that the session key should look random for an outsider adversary.

Our extension actually differs from the original PAKE security model [3] when defining the quality of an adversary. The goal of an adversary is to distinguish the answer of the Test-query on a fresh instance: a trivial attack is the so-called on-line dictionary attack which consists in trying all the possibilities when interacting with a target player. For PAKE schemes, the advantage of such an attack is q_s/N, where q_s is the number of Send-queries and N the number of possible passwords. A secure PAKE scheme should guarantee this is the best attack, or equivalently that the advantage of any adversary is bounded by $q_s \times 2^{-m}$, where m is the min-entropy of the password distribution. In our extension, for LAKE, the trivial attack consists in trying all the possibilities for $\mathsf{priv}, priv'$ with a word W in $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

Definition 3 (Security for LAKE). *A* LAKE *protocol is claimed* (t, ε)*-secure if the advantage of any adversary running in time* t *is bounded by* $q_s \times 2^{-m} \times \mathsf{Succ}^{\mathcal{L}}(t) + \varepsilon$*, where* m *is the min-entropy of the pair* $(\mathsf{priv}, \mathsf{priv}')$*, and* $\mathsf{Succ}^{\mathcal{L}}(t)$ *is the maximal success an adversary can get in finding a word in any* $\mathcal{L}_{\mathsf{pub},\mathsf{priv}}$ *within time* t*.*

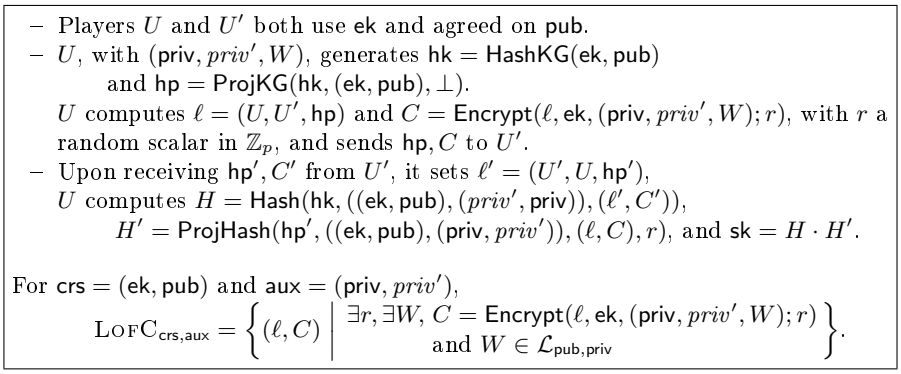

Fig. 4. One-Round LAKE

Note that the min-entropy of the pair $(\mathsf{priv}, \mathsf{priv}')$ might be conditioned to the public information from the context.

Our Instantiation. Using the same approach as Katz and Vaikuntanathan for their one-round PAKE [24], one can design the scheme proposed on Figure 4, in which both users U and U' use the encryption key ek and the public part pub. This defines $\mathsf{crs} = (\mathsf{ek}, \mathsf{pub})$. When running the protocol, U owns a word W for a private part priv, and thinks about a private part $priv'$ for U', while U' owns a word W' for a private part priv', and thinks about a private $priv$ for U.

This gives a concrete instantiation of one-round LAKE as soon as one can design a KV-SPHF on the language $\textsc{Lof}\mathrm{C}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},priv')} = \{(\ell, C) \mid \exists r, \exists W,\ C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, priv', W); r) \text{ and } W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}\}$. More precisely, each player encrypts $(\mathsf{priv}, priv', W)$ as a vector, which thus leads to $C = (C_1, C_2, C_3)$. We then use the combination of three SPHFs: two on equality-test for the plaintexts priv (for C_1) and $priv'$ (for C_2), and one on $\textsc{Lof}\mathrm{C}_{(\mathsf{ek},\mathsf{pub}),\mathsf{priv}}$ for the ciphertext C_3 of $W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

We stress that hk and hp can depend on crs but not on aux, hence the notations used in the Figure 4. Using a similar proof as in [24], one can state the following theorem (more details on the security model and the full proof can be found in the SPHF full version [6]):

Theorem 4. *If the encryption scheme is* IND-CCA*, and* $\textsc{Lof}\mathrm{C}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},priv')}$ *languages admit* KV-SPHF*s, then our* LAKE *protocol is secure.*

From LAKE to PAKE. One can remark that this theorem immediately proves the security of our PAKE from Figure 1: one uses $\mathsf{priv} = \mathsf{priv}' = \mathsf{pw}$ and $\mathsf{pub} = \emptyset$, for the language of the ciphertexts of pw.

6.2 Two-Flow Waters Blind Signature

In [9], the authors presented a technique to do efficient blind signatures using an SPHF: it is still the most efficient Waters blind signature known so far. In addition, the resulting signature is a classical Waters signature.

The construction basically consists in encrypting the message bit-by-bit under distinct bases, that will allow the generation of a masked Waters hash of the message. Thereafter, the signer will easily derive a masked signature the user will eventually unmask. However, in order to generate the masked signature, the signer wants some guarantees on the ciphertexts, namely that some ciphertexts contain a bit (in order to allow extractability) and that another ciphertext contains a Diffie-Hellman value. Using our new techniques, we essentially improve on the proof of bit encryption by using the above randomness-reuse technique.

Construction. We refer the reader to [9] for the notations and to the SPHF full version [6] for details on the proof, and also for the complete construction of the GL-SPHF. Here, we give a sketch of the protocol (in which i always ranges from 1 to ℓ, except if stated otherwise) and its communication cost:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates a pairing-friendly system $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e; g_1, g_2)$, with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively, a random generator $h_s \in \mathbb{G}_1$ as well as independent generators $\boldsymbol{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters hash function $\mathcal{F}(M) = u_0 \prod_i u_i^{M_i}$, for $M = (M_i)_i \in \{0,1\}^\ell$, and finally random scalars $(x_i)_i \in \mathbb{Z}_p^\ell$. It also sets $\mathsf{ek} = (h_i)_i = (g_1^{x_i})_i$ and $g_s = \prod_i h_i$. It outputs the global parameters $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \boldsymbol{u})$. Essentially, g_1 and ek compose the encryption key for an ElGamal ciphertext on a vector, applying the randomness-reuse technique, while g_s, g_2 and h_s are the bases used for the Waters signature;
- $\mathsf{KeyGen}(\mathsf{param})$ picks at random $x \in \mathbb{Z}_p$, sets the signing key $\mathsf{sk} = h_s^x$ and the verification key $\mathsf{vk} = (g_s^x, g_2^x)$;
- $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, M)\rangle$ runs as follows, where $\mathcal{U}$ wants to get a signature on $M = (M_i)_i \in \{0,1\}^\ell$:
  - Message Encryption: $\mathcal{U}$ chooses a random $r \in \mathbb{Z}_p$ and encrypts $u_i^{M_i}$ for all the i's with the same random r: $c_0 = g_1^r$ and $(c_i = h_i^r u_i^{M_i})_i$. $\mathcal{U}$ also encrypts vk_1^r, into $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$, with a different random s: It eventually sends $(c_0, (c_i)_i, (d_0, d_1)) \in \mathbb{G}_1^{\ell+3}$;
  - Signature Generation: $\mathcal{S}$ first computes the masked Waters hash of the message $c = u_0 \prod_i c_i = (\prod_i h_i)^r \mathcal{F}(M) = g_s^r \mathcal{F}(M)$, and generates the masked signature $(\sigma_1' = h_s^x c^t = h_s^x g_s^{rt} \mathcal{F}(M)^t, \sigma_2 = (g_s^t, g_2^t))$ for a random $t \xleftarrow{\$} \mathbb{Z}_p$;
  - SPHF: $\mathcal{S}$ needs the guarantee that each ElGamal ciphertext (c_0, c_i) encrypts either 1 or u_i under the key (g_1, h_i), and (d_0, d_1) encrypts the Diffie-Hellman value of $(g_1, c_0, \mathsf{vk}_1)$ under the key (g_1, h_1). The signer chooses a random $\mathsf{hk} = (\eta, (\theta_i)_i, (\nu_i)_i, \gamma, (\mu_i)_i, \lambda)$ and sets $\mathsf{hp}_1 = g_1^\eta \cdot \prod_i h_i^{\theta_i} \cdot \mathsf{vk}_1^\lambda$, $(\mathsf{hp}_{2,i} = u_i^{\theta_i} c_0^{\nu_i} (c_i/u_i)^{\mu_i})_i$, $(\mathsf{hp}_{3,i} = g_1^{\theta_i} h_i^{\mu_i})_i$, and $\mathsf{hp}_4 = g_1^\gamma h_1^\lambda$, then $H = c_0^\eta \cdot \prod_i c_i^{\theta_i} \cdot d_0^\gamma \cdot d_1^\lambda = \mathsf{hp}_1^r \cdot \prod_i \mathsf{hp}_{2,i}^{M_i} \cdot \mathsf{hp}_{3,i}^{-rM_i} \cdot \mathsf{hp}_4^s = H' \in \mathbb{G}_1$. This SPHF is easily obtained from the above GL-SPHF on bit encryption, as shown in the SPHF full version [6];
  - Masked Signature: $\mathcal{S}$ sends $(\mathsf{hp}, \Sigma = \sigma_1' \cdot H, \sigma_2) \in \mathbb{G}_1^{2\ell+3} \times \mathbb{G}_2$;
  - Signature Recovery: Upon receiving $(\mathsf{hp}, \Sigma, \sigma_2)$, using his witnesses and hp, $\mathcal{U}$ computes H' and unmasks σ_1'. Thanks to the knowledge of r, it can

compute $\sigma_1 = \sigma'_1 \cdot (\sigma_{2,1})^{-r}$. Note that if $H' = H$, then $\sigma_1 = h_s^x \mathcal{F}(M)^t$, which together with $\sigma_2 = (g_s^t, g_2^t)$ is a valid Waters signature on M;

– $\mathsf{Verif}(\mathsf{vk}, M, (\sigma_1, (\sigma_{2,1}, \sigma_{2,2}))$, checks whether both $e(\sigma_{2,1}, g_2) = e(g_s, \sigma_{2,2})$ and $e(\sigma_1, g_2) = e(h, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2})$ are satisfied or not.

Complexity. The whole process requires only $3\ell + 7$ elements in $\mathbb{G}_1$ ($\ell + 3$ for the ciphertexts, $2\ell + 4$ for the projection key, Σ and $\sigma_{2,1}$) and 1 in $\mathbb{G}_2$ ($\sigma_{2,2}$). This is more efficient than the instantiation from [9] ($5\ell + 6$ elements in $\mathbb{G}_1$ and 1 in $\mathbb{G}_2$) already using an SPHF, and much more efficient than the instantiation from [8] ($6\ell + 7$ elements in $\mathbb{G}_1$ and $6\ell + 5$ in $\mathbb{G}_2$) using a Groth-Sahai [21] NIZK proof.

7 Application of TSPHFs to Zero-Knowledge Arguments

In this section, we are interested in the application of SPHFs and TSPHFs to zero-knowledge arguments. Zero-knowledge arguments are used to convince a verifier that some statement or word x is in a given NP-language $\mathcal{L}$, defined by a polynomial time relation $\mathcal{R}$: $\mathcal{L} = \{x \,|\, \exists (w, y), \mathcal{R}(x, (w, y)) = 1\}$. This means that a word x is valid if there exists a witness (w, y) such that $\mathcal{R}(x, (w, y)) = 1$. The witness is divided in two parts (w, y): we want to prove that we know some w for which there exists y such that $\mathcal{R}(x, (w, y)) = 1$. We use the notation of [12] and write this as:

$$\text{Ƞ}w,\ \exists y,\ \mathcal{R}(x, (w, y)) = 1.$$

This formalism generalizes both extractable arguments of knowledge (when $y = \perp$) and non-extractable zero-knowledge arguments (when $w = \perp$). More precisely, we are interested in (partially) extractable zero-knowledge arguments (E-ZK) and extractable honest-verifier zero-knowledge arguments (HVE-ZK). E-ZK have to be *complete*, *sound*, *extractable* and *zero-knowledge*. Completeness states that an honest verifier always accepts a proof made by an honest prover for a valid statement and using a valid witness. Soundness states that no adversary can make an honest verifier accept a proof of a false statement x. Extractability states that there exists an extractor able to simulate a verifier and to output a valid partial witness w from any successful interaction with an adversary playing the role of a prover. The zero-knowledge property ensures that it is possible to simulate a prover for any true statement x even without access to a witness (w, y) for this statement x. HVE-ZK are similar to E-ZK with the difference, that for HVE-ZK, the zero-knowledge property holds only when verifiers are honest. Formal definitions can be found in the TSPHF full version [7].

We first show that SPHFs enable efficient constructions of HVE-ZK arguments. But these constructions are often not witness-indistinguishable, *i.e.*, a malicious prover may be able to distinguish which witness has been used by an honest prover[2], in general. Then we show that TSPHFs can overcome this limitation and we provide efficient constructions of E-ZK from TSPHFs.

[2] The formal definition of witness-indistinguishability can be found in the TSPHF full version [7].

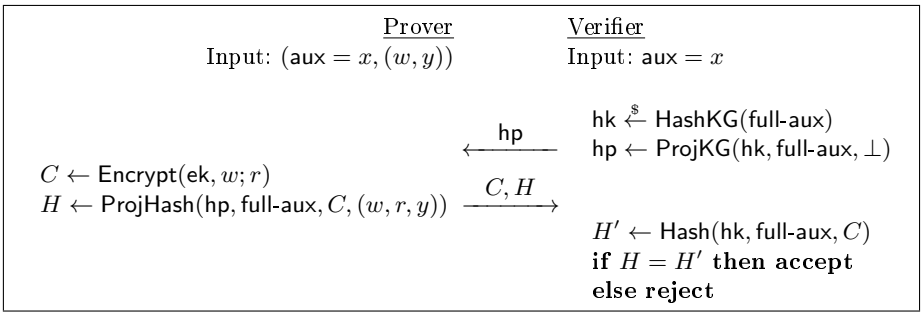

Fig. 5. Extractable Honest-Verifier Zero-Knowledge Argument from KV-SPHFs.

7.1 Honest-Verifier Zero-Knowledge Arguments from SPHFs

The idea of the construction is that a prover, who knows some valid statement x together with a valid witness (w, y), encrypts w, using an IND-CPA encryption scheme, in some ciphertext C, under some encryption key ek contained in crs. Then, using an SPHF, he shows that the ciphertext C is an encryption of a valid partial witness w for the word x: the verifier chooses some hashing key hk and sends the corresponding projection key hp to the prover; the prover sends back the hash value H of the ciphertext C computed from hp, w, y and the random coins used in C, using ProjHash; and the verifier checks he gets the same hash value from hk, using Hash. If the SPHF is a KV-SPHF, the prover can send the ciphertext C together with H after receiving hp from the verifier. This yields a two-flow protocol. More precisely, we use a KV-SPHF for the following language:

$$\text{LoFC}_{\mathsf{full\text{-}aux}} = \{C \mid \exists w,\, \exists r,\, \exists y,\; C = \mathsf{Encrypt}(\mathsf{ek}, w; r) \text{ and } \mathcal{R}(x, (w, y))\},$$

where aux is the statement x, and crs contains the encryption key ek and possibly some global parameters related to the language $\mathcal{L}$ associated with the relation $\mathcal{R}$. The complete protocol is depicted in Figure 5. In all this section, aux is public, and so it is no more required that ProjHash does not use its input aux.

It is possible to use a GL-SPHF instead of a KV-SPHF for the above language, if the ciphertext C is sent before hp. The protocol becomes three-flow but can require fewer bits to be transmitted, because GL-SPHFs are often more efficient than KV-SPHFs. Details can be found in the TSPHF full version [7].

Completeness comes from the correctness of the SPHF and soundness comes from the statistical smoothness of the SPHF. The extractor just acts as an honest verifier and decrypts the ciphertext C of the adversarial prover at the end. The simulator for the honest-verifier zero-knowledge property just encrypts an arbitrary value in C and computes H using hk: $H = \mathsf{Hash}(\mathsf{hk}, \mathsf{full\text{-}aux}, C)$. The IND-CPA property of the encryption scheme used for C ensures the simulator transcripts are computationally indistinguishable from real transcripts, and so the proposed construction is honest-verifier zero-knowledge.

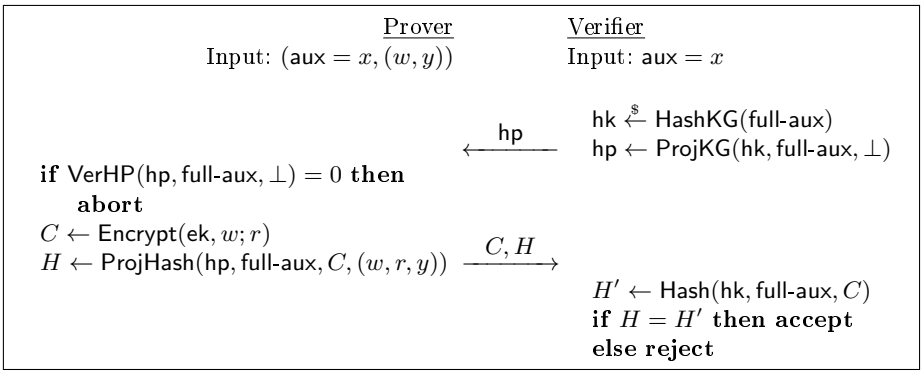

Fig. 6. Extractable Zero-Knowledge Argument from KV-TSPHFs

7.2 Limitation of SPHFs

Unfortunately, without any extra property on the SPHF, the above construction is not witness indistinguishable, and so not zero-knowledge, in general. The main problem is that, for some SPHFs, it may be possible to generate hp in such a way that the hash value H computed by the prover (through ProjHash) depends on the witness used. This happens, in particular, when the language $\text{LOFC}_{\mathsf{full\text{-}aux}}$ of the SPHF (and also the language $\mathcal{L}$ of the HVE-ZK) is a disjunction of two languages and when the generic construction of [1] for disjunctions is used to construct the SPHF.

The previous problem does not happen for SPHFs where it is easy to distinguish valid hp from invalid ones, such as all SPHFs of this article. However, even in this case, we do not see how to prove that the resulting generic construction yields a zero-knowledge argument, because, if the simulator does not have access to hk, but only to hp, there is no trivial way to compute H.

7.3 Zero-Knowledge Arguments from TSPHFs

Let us now introduce our generic two-flow construction of E-ZK arguments from TSPHFs. The scheme is depicted in Figure 6. It is similar to the above generic construction of HVE-ZK arguments from SPHFs, except the KV-SPHF is replaced by a KV-TSPHF and the verifier aborts if the received hp is not valid.

It is also possible to use a GL-TSPHF (instead of a KV-TSPHF), at the expense of requiring three flows instead of two. In addition, if the IND-CPA encryption scheme is replaced by a labeled IND-CCA encryption scheme and the TSPHF is adapted, the resulting zero-knowledge argument becomes true-simulation extractable, which roughly means that it is possible to extract the witness of any successful proof from an adversarial prover, even if this adversarial prover has access to simulated transcripts of true statements. Details can be found in the TSPHF full version [7].

Completeness and correctness can be proven as above. For the zero-knowledge property, the simulator consists in encrypting an arbitrary value in C and computing H using hp and the trapdoor τ': $\mathsf{THash}(\mathsf{hp}, \mathsf{full\text{-}aux}, C, \tau')$. This works thanks to the IND-CPA property of the encryption scheme.

Soundness and extractability are slightly more complex to prove and require that, for any w and x, knowing τ provides a way to test whether x is valid and w is a partial witness of x, with overwhelming probability. This property is actually always verified by TSPHFs constructed as in Section 4.2. Proofs are given in the TSPHF full version [7].

7.4 Instantiations and Comparison with Ω-Protocols

Let us consider the KV-SPHFs on ElGamal ciphertexts of Section 5.1, in which possibly some of the $a_{k,i}$'s, $A_{k,j}$'s and B_k's are moved from crs to aux, which is possible here, since ProjHash is now allowed to depend on aux. If we apply the generic constructions of HVE-ZK and E-ZK to these KV-SPHFs (after transforming them to KV-TSPHFs, using the generic transformation of Section 4.2, for the E-ZK construction), we get HVE-ZK and E-ZK for languages of systems of linear multi-exponentation relations in cyclic groups:

$$\mathcal{H}(X_i)_i \in \mathbb{G}^n, \exists (y_j)_j \in \mathbb{Z}_p^m, \forall k \in \{1, \ldots, t\}, \prod_{i=1}^{n} X_i^{a_{k,i}} \cdot \prod_{j=1}^{m} A_{k,j}^{y_j} = B_k,$$

where a statement x is a tuple containing all the constants $a_{k,i}$, $A_{k,j}$ and B_k, or some of them (in this case, the other constants are in crs).

Compared to Ω-protocols [18], which are the classical way to do a HVE-ZK, our HVE-ZK protocol from KV-SPHFs is two-flow instead of three-flow and has a lower communication complexity. Whereas our E-ZK protocol from TSPHFs is still two-flow instead of three-flow, it verifies a stronger notion of security (zero-knowledge versus honest-verifier zero-knowledge) and has just a slightly greater communication complexity.

Acknowledgments

We would like to thank Jonathan Katz for his helpful comments and anonymous referees of Crypto 2013 for their valuable inputs.

This work was supported in part by the French ANR-12-INSE-0014 SIMPATIC Project and in part by the European Commission through the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet. The second author was funded by a Sofja Kovalevskaja Award of the Alexander von Humboldt Foundation and the German Federal Ministry for Education and Research.

References

1. Abdalla, M., Chevalier, C., Pointcheval, D.: Smooth projective hashing for conditionally extractable commitments. In: Halevi, S. (ed.) Advances in Cryptology

– CRYPTO 2009. Lecture Notes in Computer Science, vol. 5677, pp. 671–689. Springer (Aug 2009)
2. Bellare, M., Boldyreva, A., Staddon, J.: Randomness re-use in multi-recipient encryption schemes. In: Desmedt, Y. (ed.) PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 2567, pp. 85–99. Springer (Jan 2003)
3. Bellare, M., Pointcheval, D., Rogaway, P.: Authenticated key exchange secure against dictionary attacks. In: Preneel, B. (ed.) Advances in Cryptology – EUROCRYPT 2000. Lecture Notes in Computer Science, vol. 1807, pp. 139–155. Springer (May 2000)
4. Bellovin, S.M., Merritt, M.: Encrypted key exchange: Password-based protocols secure against dictionary attacks. In: 1992 IEEE Symposium on Security and Privacy. pp. 72–84. IEEE Computer Society Press (May 1992)
5. Ben Hamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: Efficient uc-secure authenticated key-exchange for algebraic languages. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography. Lecture Notes in Computer Science, vol. 7778, pp. 272–291. Springer (2013)
6. Benhamouda, F., Blazy, O., Chevalier, C., Pointcheval, D., Vergnaud, D.: New smooth projective hash functions and one-round authenticated key exchange. Cryptology ePrint Archive, Report 2013/034 (2013), `http://eprint.iacr.org/`
7. Benhamouda, F., Pointcheval, D.: Trapdoor smooth projective hash functions. Cryptology ePrint Archive, Report 2013/341 (2013), `http://eprint.iacr.org/`
8. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 6571, pp. 403–422. Springer (Mar 2011)
9. Blazy, O., Pointcheval, D., Vergnaud, D.: Round-optimal privacy-preserving protocols with smooth projective hash functions. In: Cramer, R. (ed.) TCC 2012: 9th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 7194, pp. 94–111. Springer (Mar 2012)
10. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004. Lecture Notes in Computer Science, vol. 3152, pp. 41–55. Springer (Aug 2004)
11. Camenisch, J., Casati, N., Groß, T., Shoup, V.: Credential authenticated identification and key exchange. In: Rabin, T. (ed.) Advances in Cryptology – CRYPTO 2010. Lecture Notes in Computer Science, vol. 6223, pp. 255–276. Springer (Aug 2010)
12. Camenisch, J., Krenn, S., Shoup, V.: A framework for practical universally composable zero-knowledge protocols. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 449–467. Springer (Dec 2011)
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd Annual Symposium on Foundations of Computer Science. pp. 136–145. IEEE Computer Society Press (Oct 2001)
14. Canetti, R., Halevi, S., Katz, J., Lindell, Y., MacKenzie, P.D.: Universally composable password-based key exchange. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 404–421. Springer (May 2005)

15. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) Advances in Cryptology – CRYPTO'98. Lecture Notes in Computer Science, vol. 1462, pp. 13–25. Springer (Aug 1998)
16. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology – EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 45–64. Springer (Apr / May 2002)
17. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
18. Garay, J.A., MacKenzie, P.D., Yang, K.: Strengthening zero-knowledge protocols using signatures. Journal of Cryptology 19(2), 169–209 (Apr 2006)
19. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610 (2012), `http://eprint.iacr.org/`
20. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 524–543. Springer (May 2003), `http://eprint.iacr.org/2003/032.ps.gz`
21. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) Advances in Cryptology – EUROCRYPT 2008. Lecture Notes in Computer Science, vol. 4965, pp. 415–432. Springer (Apr 2008)
22. Jutla, C.S., Roy, A.: Relatively-sound NIZKs and password-based key-exchange. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 7293, pp. 485–503. Springer (May 2012)
23. Katz, J., Ostrovsky, R., Yung, M.: Efficient password-authenticated key exchange using human-memorable passwords. In: Pfitzmann, B. (ed.) Advances in Cryptology – EUROCRYPT 2001. Lecture Notes in Computer Science, vol. 2045, pp. 475–494. Springer (May 2001)
24. Katz, J., Vaikuntanathan, V.: Round-optimal password-based authenticated key exchange. In: Ishai, Y. (ed.) TCC 2011: 8th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 6597, pp. 293–310. Springer (Mar 2011)
25. Libert, B., Yung, M.: Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In: Cramer, R. (ed.) TCC 2012: 9th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 7194, pp. 75–93. Springer (Mar 2012)
26. Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd Annual ACM Symposium on Theory of Computing. ACM Press (May 1990)
27. Pointcheval, D.: Password-based authenticated key exchange (invited talk). In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography. Lecture Notes in Computer Science, vol. 7293, pp. 390–397. Springer (May 2012)
28. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science. pp. 543–553. IEEE Computer Society Press (Oct 1999)
29. Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 114–127. Springer (May 2005)