This is the Full Version of the part on Smooth Projective Hash Functions
of the Extended Abstract "New Techniques for SPHFs and Efficient One-Round PAKE Protocols"
that appears in Advances in Cryptology – Proceedings of CRYPTO 2013 (18–22 august 2013, Santa Barbara, California, USA)
R.Canetti and J. A. Garay Eds. Springer-Verlag, LNCS 8042, pages 449–475.

New Smooth Projective Hash Functions and One-Round Authenticated Key Exchange

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) has received deep attention in the last few years, with a recent improvement by Katz and Vaikuntanathan, and their one-round protocols: the two players just have to send simultaneous flows to each other, that depend on their own passwords only, to agree on a shared high entropy secret key. To this aim, they followed the Gennaro and Lindell's approach, with a new kind of *Smooth-Projective Hash Functions* (SPHFs). They came up with the first concrete one-round PAKE, secure in the Bellare, Pointcheval, and Rogaway's model, but at the cost of a simulation-sound NIZK, which makes the overall construction not really efficient.

This paper follows their path with a new efficient instantiation of SPHF on Cramer-Shoup ciphertexts. It then leads to the design of the most efficient PAKE known so far: a one-round PAKE with two simultaneous flows consisting of 6 group elements each only, in any DDH-group without any pairing. We thereafter show a generic construction for SPHFs, in order to check the validity of complex relations on encrypted values. This allows to extend this work on PAKE to the more general family of protocols, termed *Langage-Authenticated Key Exchange* (LAKE) by Ben Hamouda, Blazy, Chevalier, Pointcheval, and Vergnaud, but also to blind signatures. We indeed provide the most efficient blind Waters' signature known so far.

Keywords. Authenticated Key Exchange, Blind Signatures, Smooth Projective Hash Functions

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 agree on over the phone, hence PAKE, for *Password-Authenticated Key Exchange*. The most famous instantiation has been proposed by Bellovin and Merritt [BM92], EKE for Encrypted Key Exchange, which simply consists of a Diffie-Hellman key exchange [DH76], 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 [BPR00] (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 [Poi12]. Katz, Ostrovsky and Yung [KOY01] 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 [GL03] (GL), making use of smooth projective hash functions.

Smooth Projective Hash Functions (SPHFs) were introduced by Cramer and Shoup [CS02] 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 [CS98]. They can be

* CNRS – UMR 8548 and INRIA – EPI Cascade

seen as a kind of implicit designated-verifier proofs of membership [ACP09, BPV12]. Basically, SPHFs are families of pairs of functions (Hash, ProjHash) defined on a language L. These functions are indexed by a pair of associated keys (hk, 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 [GL03] (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.* [CHK+05], but for static corruptions only. It has been improved by Abdalla, Chevalier and Pointcheval [ACP09] 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 [KV11] (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. 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 [NY90], with an ElGamal-like encryption scheme and a *simulation-sound non-interactive zero-knowledge* (SS-NIZK) proof [Sah99]. Such an SS-NIZK proof is quite costly in general. They suggested to use Groth-Sahai [GS08] proofs in bilinear groups and the linear encryption [BBS04] 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 [LY12] 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 [JR12] 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.

Language-Authenticated Key Exchange. A generalization of AKE protocols has been recently proposed, so-called *Language-Authenticated Key Exchange* (LAKE) [BBC+13]: it allows two users, Alice and Bob, each owning a word in a specific language, to agree on a shared high entropy secret if each user knows a word in the language the other thinks about. More precisely, they first both agree on public parameters pub, Bob will think about *priv* for his expected Alice's value of priv, Alice will do the same with *priv'* for Bob's private value priv'; eventually, if *priv* = priv and *priv'* = priv', and if they both know words in the appropriate languages, then the key agreement will succeed. In case of failure, no information should leak to the players about the reason of failure, except that the inputs did not satisfy the relations, or the languages were not consistent. Eavesdroppers do not even learn the outcome.

This formalism encompasses PAKE, and their first construction follows the GL approach for PAKE: each player commits to the private values (his own value priv, and his expected partner's value *priv'*) as well as his own word, and projection keys are sent to compute random values that will be the same if and only if everything is consistent. To achieve one-round LAKE, one also needs KV-SPHF on ciphertexts for plaintext-equality tests (equality of the private values and expected private values) and for language-membership.

Achievements. Our main 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 [KV11], 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 [BBC+13]. 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 [CCGS10]. Second, thanks to a new GL-SPHF, we improve on the *blind signature* scheme presented in [BPV12], 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 [Wat05]. Our protocol is round-optimal, since it consists of two flows, and leads to a classical short Waters signature.

As a side contribution, we introduce a new generic framework to construct SPHFs aiming at making easier the construction and the proof of SPHFs on complex languages. Using this framework, we were able to construct SPHFs for any language handled by the Groth-Sahai NIZK proofs, and so for any $\mathcal{NP}$-language.

Outline of the Paper. In Section 2, we first revisit the different definitions for SPHFs proposed in [CS02, GL03, KV11], denoted respectively CS-SPHFs, GL-SPHFs and KV-SPHFs. While CS-SPHF was 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 a KV-SPHF for our PAKE/LAKE application and a GL-SPHF for the blind signature. In Section 2.4, we introduce our main contribution, the construction of a KV-SPHF on Cramer-Shoup ciphertexts. This KV-SPHF leads to the construction of our efficient one-round PAKE in Section 2.5. In Section 3, we present a simplified version of our generic framework (fully described in Appendix D). We then show our efficient SPHFs on multi-exponentiation equations and on bit encryption, without pairings, in Section 4. Finally, in Section 5, we introduce our two other constructions based on these SPHFs: our one-round LAKE and our blind signature scheme.

2 New SPHF on Cramer-Shoup Ciphertexts and PAKE

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 [CS02] type (CS-SPHF) introduced for enhancing an IND-CPA encryption scheme to IND-CCA, the Gennaro and Lindell [GL03] type (GL-SPHF) introduced for PAKE, and the Katz and Vaikuntanathan [KV11] 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 [KV11], 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. For that, we use the classical notion of statistical distance recalled in Appendix A.2.

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 [BBC+13], 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 $(\mathsf{crs}, \mathsf{aux})$: 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: 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). The CS labeled encryption scheme is recalled in Appendix A.3. We briefly review it here. 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 [GL03] proposed an SPHF on labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^4$. 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,

- 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

one can compute the hash value in two different ways, for the language $\textsc{Lofc}_{\mathsf{ek},m}$ 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 $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \nu) \xleftarrow{\$} \mathbb{Z}_p^5$; 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 $\textsc{Lofc}_{\mathsf{ek},m}$ 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 D.3 as an illustration of our new framework.

2.5 An Efficient One-Round PAKE

Review of the Katz and Vaikuntanathan's PAKE. As explained earlier, Katz and Vaikuntanathan recently proposed a one-round PAKE in [KV11]. Their general framework follows Gennaro and Lindell [GL03] approach: each player sends an encryption of the password, and then uses an SPHF on the partner's ciphertext to check whether it actually contains the same password. The two hash values are multiplied to produce the session key. If the encrypted passwords are the same, the different ways to compute the hash values (Hash and ProjHash) give the same results. If the passwords differ, the smoothness makes the values computed by each player independent. To this aim, the authors need 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.

Our Construction. Our KV-SPHF on Cramer-Shoup ciphertexts can be used in the Katz and Vaikuntanathan framework for PAKE [KV11]. 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 [GS08], or 20 group elements for the Jutla and Roy's [JR12] improvement using a relatively-sound NIZK.

The formal security result follows from the Theorem 4 in Section 5.1. 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. Under the DLin assumption (which is a weaker assumption in any group), still without requiring pairing-friendly groups, our construction would make each user to send 9 group elements only.

3 Generic Framework for SPHFs

3.1 Introduction

In Appendix D, we propose a formal framework for SPHFs using a new notion of graded rings, derived from [GGH12]. 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 [BBC+13] for any language handled by the Groth-Sahai NIZK proofs, and so for any $\mathcal{NP}$-language (see Section 5.1).

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.

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

3.2 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 $n = 5$:

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

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.

3.3 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 Appendix D 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 $\Gamma(C) \odot \boldsymbol{\alpha} = \mathsf{hp}$.

4 Concrete Constructions of SPHFs

In this section, we illustrate more our generic framework, by constructing more evolved SPHFs without pairings. More complex constructions of SPHFs, namely for any language handled by the Groth-Sahai NIZK proofs, are detailed in Appendix D.

4.1 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 [BBC+13]. 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.

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 $\mathsf{ek} = (g, h = g^x)$. We are interested in languages on the ciphertexts $C_{1,i} = (u_{1,i} = g^{r_{1,i}}, e_{1,i} = h^{r_{1,i}} \cdot X_i)$, for $X_1, \ldots, X_{n_1} \in \mathbb{G}$, and $C_{2,j} = (u_{2,j} = g^{r_{2,j}}, e_{2,j} = h^{r_{2,j}} \cdot g^{y_j})$, for $y_1, \ldots, y_{n_2} \in \mathbb{Z}_p$, such that:

$$\prod_{i=1}^{n_1} X_i^{a_i} \cdot \prod_{j=1}^{n_2} A_j^{y_j} = B, \text{ with } \begin{array}{l} \mathsf{crs} = (p, \mathbb{G}, \mathsf{ek}, A_1, \ldots, A_{n_2}) \\ \mathsf{aux} = (a_1, \ldots, a_{n_1}, B) \in \mathbb{Z}_p^{n_1} \times \mathbb{G}. \end{array} \tag{1}$$

We insist that, here, the elements $(A_1, \ldots, A_{n_2}) \in \mathbb{G}^{n_2}$ are known in advance, contrarily to equation (2) in Appendix D.4, where they are in aux and make the SPHF to use pairings.

In the following, i and j will always range from 1 to n_1 and from 1 to n_2 respectively in all the products $\prod_i$, $\prod_j$ and tuples $(\cdot)_i$, $(\cdot)_j$. We can define the following elements, and namely the $(2n_2+1, 2n_2+2)$-matrix Γ that uses the knowledge of the elements $(A_j)_j$:

$$\Gamma = \left(\begin{array}{c|ccc|ccc|c} g & 1 & \ldots & 1 & 1 & \ldots & 1 & h \\ \hline 1 & g & & 1 & h & & 1 & 1 \\ \vdots & & \ddots & & & \ddots & & \vdots \\ 1 & 1 & & g & 1 & & h & 1 \\ \hline 1 & & & & g & & 1 & A_1^{-1} \\ \vdots & & 1 & & & \ddots & & \vdots \\ 1 & & & & 1 & & g & A_{n_2}^{-1} \end{array}\right)$$

$$\begin{aligned} \Theta_{\mathsf{aux}}(\boldsymbol{C}) &= \left(\textstyle\prod_i u_{1,i}^{a_i}, (u_{2,j})_j, (e_{2,j})_j, \prod_i e_{1,i}^{a_i} / B\right) \\ \boldsymbol{\lambda} &= \left(\textstyle\sum_i a_i r_{1,i}, (r_{2,j})_j, (y_j)_j\right) \\ \boldsymbol{\lambda} \odot \Gamma &= \left(g^{\sum_i a_i r_{1,i}}, (g^{r_{2,j}})_j, (h^{r_{2,j}} \cdot g^{y_j})_j, h^{\sum_i a_i r_{1,i}} / \textstyle\prod_j A_j^{y_j}\right) \end{aligned}$$

We recall that in the matrix, 1 is the neutral element in $\mathbb{G}$ and can thus be ignored. When one considers the discrete logarithms, they become 0, and thus the matrix is triangular. The three diagonal blocks impose

the value of $\boldsymbol{\lambda}$, and the last column defines the relation: the last component of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$ is $\prod_i e_{1,i}^{a_i}/B = h^{\sum_i a_i r_{1,i}} \cdot \prod_i X_i^{a_i}/B$, which is equal to the last component of $\boldsymbol{\lambda} \odot \Gamma = h^{\sum_i a_i r_{1,i}} / \prod_j A_j^{y_j}$, if and only if the relation (1) is satisfied. It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $\mathsf{hp}_1 = g^\eta h^\nu$, $(\mathsf{hp}_{2,j} = g^{\theta_j} h^{\mu_j})_j$, and $(\mathsf{hp}_{3,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, (\theta_j)_j, (\mu_j)_j, \nu)$:

$$H = \prod_i (u_{1,i}^\eta e_{1,i}^\nu)^{a_i} \cdot \prod_j (u_{2,j}^{\theta_j} e_{2,j}^{\mu_j})/B^\nu = \mathsf{hp}_1^{\sum_i a_i r_{1,i}} \cdot \prod_j (\mathsf{hp}_{2,j}^{r_{2,j}} \cdot \mathsf{hp}_{3,j}^{y_j}) = H'.$$

As a consequence, the ciphertexts and the projection keys (which have to be exchanged in a protocol) globally consist of $2n_1 + 4n_2 + 1$ elements from $\mathbb{G}$.

Ciphertexts with Randomness Reuse. A first improvement consists in using multiple independent encryption keys for encrypting the y_j's: $\mathsf{ek}_{2,j} = (g, h_{2,j} = g^{x_{2,j}})$, for $j = 1, \ldots, n_2$. This allows to reuse the same random coins [BBS03]. We are interested in languages on the ciphertexts $(C_{1,i} = (u_{1,i} = g^{r_{1,i}}, e_{1,i} = h^{r_{1,i}} \cdot X_i))_i$, for $(X_i)_i \in \mathbb{G}^{n_1}$, with $(r_{1,i})_i \in \mathbb{Z}_p^{n_1}$, and $C_2 = (u_2 = g^{r_2}, (e_{2,j} = h_{2,j}^{r_2} \cdot g^{y_j})_j)$, for $(y_j)_j \in \mathbb{Z}_p^{n_2}$, still satisfying the same relation (1). This improves on the length of the ciphertexts, from $2n_1 + 2n_2$ group elements in $\mathbb{G}$ to $2n_1 + n_2 + 1$. The matrix Γ can then be compressed into:

$$\Gamma = \left(\begin{array}{c|c|ccc|c} g & 1 & 1 & \ldots & 1 & h \\ \hline 1 & g & h_{2,1} & \ldots & h_{2,n_2} & 1 \\ \hline 1 & 1 & g & & 1 & A_1^{-1} \\ \vdots & \vdots & & \ddots & & \vdots \\ 1 & 1 & 1 & & g & A_{n_2}^{-1} \end{array}\right) \qquad \begin{aligned} \Theta_{\mathsf{aux}}(\boldsymbol{C}) &= \left(\textstyle\prod_i u_{1,i}^{a_i}, u_2, (e_{2,j})_j, \prod_i e_{1,i}^{a_i}/B\right) \\ \boldsymbol{\lambda} &= (\textstyle\sum_i a_i r_{1,i}, r_2, (y_j)_j) \\ \boldsymbol{\lambda} \odot \Gamma &= (g^{\sum_i a_i r_{1,i}}, g^{r_2}, (h_{2,j}^{r_2} g^{y_j})_j, h^{\sum_i a_i r_{1,i}} / \textstyle\prod_j A_j^{y_j}) \end{aligned}$$

where again, because of the diagonal blocks in Γ, $\boldsymbol{\lambda}$ is implied by all but last components of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$. The last component of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$ is then $\prod_i e_{1,i}^{a_i}/B = \prod_i h^{a_i r_{1,i}} X_i^{a_i}/B$ and thus equal to the last component of $\boldsymbol{\lambda} \odot \Gamma$, multiplied by $\prod_i X_i^{a_i} \cdot \prod_j A_j^{y_j}/B$ that is equal to 1 if and only if the relation (1) is satisfied. It thus leads to the following $\mathsf{KV\text{-}SPHF}$, with $(\mathsf{hp}_1 = g^\eta h^\nu$, $\mathsf{hp}_2 = g^\theta \cdot \prod_j h_{2,j}^{\mu_j}$, and $(\mathsf{hp}_{3,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, \theta, (\mu_j)_j, \nu)$:

$$H = \prod_i (u_{1,i}^\eta e_{1,i}^\nu)^{a_i} \cdot \prod_j e_{2,j}^{\mu_j} \cdot u_2^\eta / B^\nu = \mathsf{hp}_1^{\sum_i a_i r_{1,i}} \cdot \mathsf{hp}_2^{r_2} \cdot \prod_j \mathsf{hp}_{3,j}^{y_j} = H'.$$

Globally, the ciphertexts and the projection keys consist of $2n_1 + 2n_2 + 3$ elements from $\mathbb{G}$. This has to be compared with $2n_1 + 4n_2 + 1$ elements from $\mathbb{G}$ in the previous construction.

Moving all the constant values from aux to crs. In some cases, all the constant values, A_j and a_i can be known in advance and public. The matrix Γ can then exploit their knowledge. We apply the randomness-reuse technique for the whole ciphertext, for both $(X_i)_i$ and $(y_j)_j$, with independent encryption keys $(h_{1,i})_i$ and $(h_{2,j})_j$ in $\mathbb{G}$. A unique random r produces $u = g^r$, and $(e_{1,i})_i$ and $(e_{2,j})_j$. This reduces the length of the ciphertext to $n_1 + n_2 + 1$ group elements in $\mathbb{G}$, but also the size of the matrix Γ:

$$\Gamma = \left(\begin{array}{c|ccc|c} g & h_{2,1} & \ldots & h_{2,n_2} & \prod_i h_{1,i}^{a_i} \\ \hline 1 & g & & 1 & A_1^{-1} \\ \vdots & & \ddots & & \vdots \\ 1 & 1 & & g & A_{n_2}^{-1} \end{array}\right) \qquad \begin{aligned} \Theta_{\mathsf{aux}}(\boldsymbol{C}) &= \left(u, (e_{2,j})_j, \textstyle\prod_i e_{1,i}^{a_i}/B\right) \qquad \boldsymbol{\lambda} = (r, (y_j)_j) \\ \boldsymbol{\lambda} \odot \Gamma &= (g^r, (h_{2,j}^r g^{y_j})_j, \textstyle\prod_i h_{1,i}^{a_i r} / \prod_j A_j^{y_j}) \end{aligned}$$

Projection keys become more compact, with only $n_2 + 1$ group elements in $\mathbb{G}$: $\mathsf{hp}_1 = g_1^\eta \cdot \prod_j h_{2,j}^{\mu_j} \cdot (\prod_i h_{1,i}^{a_i})^\nu$, and $(\mathsf{hp}_{2,j} = g^{\mu_j} A_j^{-\nu})_j$, for $\mathsf{hk} = (\eta, (\mu_j)_j, \nu)$: $H = u^\eta \cdot \prod_i e_{1,i}^{\nu a_i} \cdot \prod_j e_{2,j}^{\mu_j} / B^\nu = \mathsf{hp}_1^r \cdot \prod_j \mathsf{hp}_{2,j}^{y_j} = H'$. Globally, the ciphertexts and the projection keys consist of $n_1 + 2n_2 + 2$ elements from $\mathbb{G}$.

4.2 From ElGamal to Cramer-Shoup Encryption

In order to move from ElGamal ciphertexts to Cramer-Shoup ciphertexts, 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 blocks Γ and smallers $(\Gamma_k)_k$ for every ciphertext $(u_k, u'_k, e_k, v_k)_k$, where

$$\Gamma_k = \begin{pmatrix} g & 1 & g' & c \\ 1 & g & 1 & d \end{pmatrix} \qquad \boldsymbol{\lambda_k} = (r_k, r_k\xi_k) \qquad \begin{array}{l} \Theta_M(C_k) = (u_k, u_k^{\xi_k}, u'_k, v_k) \\ \boldsymbol{\lambda}_k \odot \Gamma_k = (g^{r_k}, g^{r_k\xi_k}, g'^{r_k}, (cd^{\xi_k})^{r_k}) \end{array}$$

The initial matrix Γ guarantees the relations on the ElGamal pairs (u_k, e_k), and the matrices Γ_k add the internal relations on the Cramer-Shoup ciphertexts. In the worst case, hk is increased by $4n$ scalars and hp by $2n$ group elements, for n ciphertexts. But some more compact matrices can be obtained in many cases, with much shorter hashing and projection keys, by merging some lines or columns in the global matrix. But this is a case by case optimization.

4.3 Generalizations

The SPHF constructions from this section are all done without requiring any pairing, but are still KV-SPHF, allowing us to handle non-quadratic multi-exponentiation equations without pairings. To further extend our formalism, we describe in the next section a concrete application to blind signatures (while with a GL-SPHF), and we present more languages in Appendix D.4.

However, as above for Cramer-Shoup ciphertexts, if one wants to satisfy several equations at a time, one just has to first consider them independently and to make a global matrix with each sub-language-matrix in a block on the diagonal. The hashing keys and the projection keys are then concatenated, and the hash values are simply multiplied. Optimizations can be possible, as shown in Appendix C for the SPHF involved in the blind signature.

4.4 GL-SPHF on Bit Encryption

As shown in Appendix D, our general framework allows to construct KV-SPHFs for any language handled by the Groth-Sahai NIZK proofs. But, 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:

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

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 $\mathsf{hk} = (\nu, \theta, \eta, \lambda)$: $H = u^\nu e^\theta = \mathsf{hp}_1^r \cdot \mathsf{hp}_2^y / \mathsf{hp}_3^{ry} = H'$.

5 More Applications of SPHFs

5.1 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 [BBC+13]. To this aim, we provide a new security model, inspired from BPR [BPR00] 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 [BBC+13] 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 [BBC+13] for more applications of LAKE.

A New Security Model for LAKE. The first security model for LAKE [BBC+13] has been given in the UC framework [Can01], as an extension of the UC security for PAKE [CHK+05]. In this paper, we propose an extension of the PAKE security model presented by Bellare, Pointcheval, and Rogaway [BPR00] 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^s_U 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 [BPR00] 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*.*

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 [KV11], one can design the scheme proposed on Figure 2, 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 $\textsc{Lofc}_{(\mathsf{ek},\mathsf{pub}),(\mathsf{priv},priv')} = \{(\ell, C) \mid \exists r, \exists W,\ C = \mathsf{Encrypt}(\ell, \mathsf{ek}, (\mathsf{priv}, priv', W); r)$ 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 $\textsc{Lofc}_{(\mathsf{ek},\mathsf{pub}),\mathsf{priv}}$ for the ciphertext C_3 of $W \in \mathcal{L}_{\mathsf{pub},\mathsf{priv}}$.

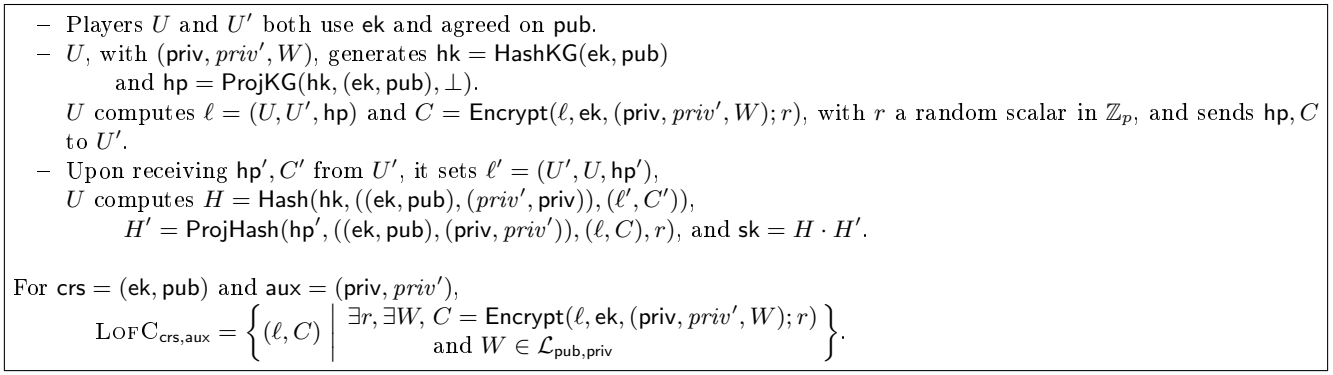

Fig. 2. One-Round LAKE

We stress that hk and hp can depend on crs but not on aux, hence the notations used in the Figure 2. Using a similar proof as in [KV11], one can state the following theorem (more details on the security model and the full proof can be found in Appendix B):

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.

5.2 Two-Flow Waters Blind Signature

Blind signature schemes, introduced by Chaum in 1982 [Cha83], allow a person to get a signature by another party without revealing any information about the message being signed. A blind signature can then be publicly verified using the unblinded message.

In [BPV12], 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 (see Appendix C.1 for the definition of Waters signatures).

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.

Definition. Before showing our new construction, let us first recall the definition of blind signatures.

A blind signature scheme $\mathcal{BS}$ is defined by three algorithms $(\mathsf{BSSetup}, \mathsf{BSKeyGen}, \mathsf{BSVerif})$ and one interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S},\mathcal{U}\rangle$:

– $\mathsf{BSSetup}(1^{\mathfrak{K}})$, generates the global parameters param of the system;
– $\mathsf{BSKeyGen}(\mathsf{param})$ is a probabilistic polynomial-time algorithm that generates a pair of keys $(\mathsf{vk}, \mathsf{sk})$ where vk is the public (verifying) key and sk is the secret (signing) key;
– $\mathsf{BSProtocol}\langle\mathcal{S}(\mathsf{sk}),\mathcal{U}(\mathsf{vk}, M)\rangle$: this is a probabilistic polynomial-time interactive protocol between the algorithms $\mathcal{S}(\mathsf{sk})$ and $\mathcal{U}(\mathsf{vk}, M)$, for a message $M \in \{0,1\}^n$. It generates a signature σ on M under vk related to sk for the user.
– $\mathsf{BSVerif}(\mathsf{vk}, M, \sigma)$ is a deterministic polynomial-time algorithm which outputs 1 if the signature σ is valid with respect to m and vk, 0 otherwise.

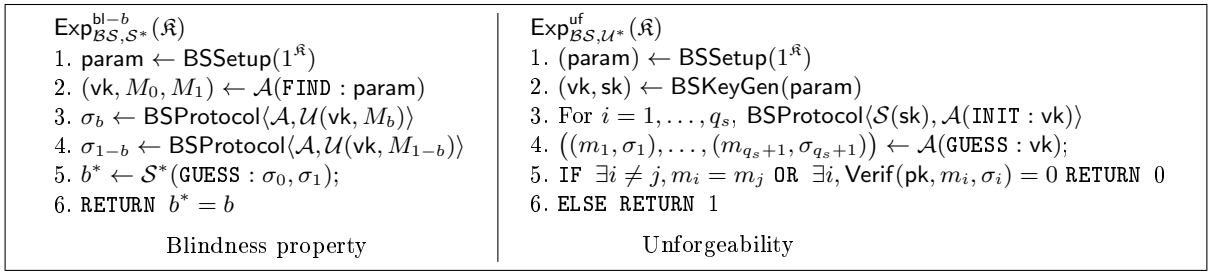

Fig. 3. Security Games for $\mathcal{BS}$

A blind signature scheme $\mathcal{BS}$ should satisfy the two following security notions: the blindness condition that is a guarantee for the signer, and the unforgeability that is a guarantee for the signer. The blindness states that a malicious signer should not be able to link the final signatures output by the user to the individual *valid* interactions with the user. We insist on *valid* executions which end with a valid signature σ of the message used by $\mathcal{U}$ under the key vk. The signer could of course send a wrong answer which would lead to an invalid signature in one execution of $\mathsf{BSProtocol}\langle \mathcal{S}(\mathsf{sk}), \mathcal{U}(\mathsf{vk}, M)\rangle$. Then, it could easily distinguish a valid signature from an invalid one, and thus valid execution of the protocol and the invalid one. However this malicious behaviour is a kind of denial of service and is out of scope of this work. Therefore, in this paper *blindness* formalizes that one valid execution is indistinguishable for the signer from other valid executions. This notion was formalized in [HKKL07] and termed *a posteriori blindness*. The unforgeability property insures that an adversary, interacting freely with an honest signer, should not be able to produce $q+1$ valid signatures after at most q complete interactions with the honest signer (for any q polynomial in the security parameter).

These security notions are precised by the security games presented on Figure 3, where the adversary keeps some internal state between the various calls `INIT`, `FIND` and `GUESS`.

Construction. 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 Appendix C;

- 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.

Security Proof. The security proof is similar to the one in [BPV12] and is given in Appendix C.2.

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 [BPV12] ($5\ell + 6$ elements in $\mathbb{G}_1$ and 1 in $\mathbb{G}_2$) already using an SPHF, and much more efficient than the instantiation from [BFPV11] ($6\ell + 7$ elements in $\mathbb{G}_1$ and $6\ell + 5$ in $\mathbb{G}_2$) using a Groth-Sahai [GS08] NIZK proof.

References

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer, August 2009.

BBC+13. Fabrice Ben Hamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient uc-secure authenticated key-exchange for algebraic languages. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013: 16th International Conference on Practice and Theory in Public-Key Cryptography*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291. Springer, 2013.

BBS03. Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemes. In Yvo Desmedt, editor, *PKC 2003: 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 85–99. Springer, January 2003.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, August 2004.

BFPV11. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Workshop on Theory and Practice in Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 403–422. Springer, March 2011.

BM92. Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *1992 IEEE Symposium on Security and Privacy*, pages 72–84. IEEE Computer Society Press, May 1992.

BPR00. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In Bart Preneel, editor, *Advances in Cryptology – EUROCRYPT 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer, May 2000.

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111. Springer, March 2012.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd Annual Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, October 2001.

CCGS10. Jan Camenisch, Nathalie Casati, Thomas Groß, and Victor Shoup. Credential authenticated identification and key exchange. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 255–276. Springer, August 2010.

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

CHK+05. Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 404–421. Springer, May 2005.

CS98. Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology – CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer, August 1998.

CS02. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, April / May 2002.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

GGH12. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610, 2012. `http://eprint.iacr.org/`.

GL03. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In Eli Biham, editor, *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 524–543. Springer, May 2003. `http://eprint.iacr.org/2003/032.ps.gz`.

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, April 2008.

HK08. Dennis Hofheinz and Eike Kiltz. Programmable hash functions and their applications. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, August 2008.

HKKL07. Carmit Hazay, Jonathan Katz, Chiu-Yuen Koo, and Yehuda Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In Salil P. Vadhan, editor, *TCC 2007: 4th Theory of Cryptography Conference*, volume 4392 of *Lecture Notes in Computer Science*, pages 323–341. Springer, February 2007.

JR12. Charanjit S. Jutla and Arnab Roy. Relatively-sound NIZKs and password-based key-exchange. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 485–503. Springer, May 2012.

KOY01. Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In Birgit Pfitzmann, editor, *Advances in Cryptology – EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer, May 2001.

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer, March 2011.

LY12. Benoît Libert and Moti Yung. Non-interactive CCA-secure threshold cryptosystems with adaptive security: New framework and constructions. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 75–93. Springer, March 2012.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*. ACM Press, May 1990.

Poi12. David Pointcheval. Password-based authenticated key exchange (invited talk). In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Workshop on Theory and Practice in Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 390–397. Springer, May 2012.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th Annual Symposium on Foundations of Computer Science*, pages 543–553. IEEE Computer Society Press, October 1999.

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

A Preliminaries

A.1 Formal Definitions of the Basic Primitives

We first recall the definitions of some of the basic tools, with the corresponding security notions and their respective success/advantage.

Hash Function Family. A hash function family $\mathcal{H}$ is a family of functions $\mathfrak{H}_K$ from $\{0,1\}^*$ to a fixed-length output, either $\{0,1\}^{\mathfrak{K}}$ or $\mathbb{Z}_p$. Such a family is said *collision-resistant* if for any adversary $\mathcal{A}$ on a random function $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$, it is hard to find a collision. More precisely, we denote

$$\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A}) = \Pr[\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}, (m_0, m_1) \leftarrow \mathcal{A}(\mathfrak{H}_K) : \mathfrak{H}_K(m_0) = \mathfrak{H}_K(m_1)], \qquad \mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(t) = \max_{\mathcal{A} \leq t}\{\mathsf{Succ}^{\mathsf{coll}}_{\mathcal{H}}(\mathcal{A})\},$$

where the latter notation means the maximum over the adversaries running within time t.

Labeled Encryption Scheme. A labeled public-key encryption scheme $\mathcal{E}$ is defined by four algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{KeyGen}(\mathsf{param})$ generates a pair of keys, the encryption key ek and the decryption key dk;
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)$ produces a ciphertext c on the input message $m \in \mathcal{M}$ under the label ℓ and encryption key ek, using the random coins r;
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, c)$ outputs the plaintext m encrypted in c under the label ℓ, or $\perp$ for an invalid ciphertext.

An encryption scheme $\mathcal{E}$ should satisfy the following properties

- *Correctness*: for all key pair $(\mathsf{ek}, \mathsf{dk})$, any label ℓ, all random coins r and all messages m,

$$\mathsf{Decrypt}(\ell, \mathsf{dk}, \mathsf{Encrypt}(\ell, \mathsf{ek}, m; r)) = m.$$

- *Indistinguishability under chosen-ciphertext attacks*: this security notion can be formalized by the following security game, where the adversary $\mathcal{A}$ keeps some internal state between the various calls `FIND` and `GUESS`, and makes use of the oracle $\mathsf{ODecrypt}$:

  - $\mathsf{ODecrypt}(\ell, c)$: This oracle outputs the decryption of c under the label ℓ and the challenge decryption key dk. The input queries (ℓ, c) are added to the list $\mathcal{CT}$.

  The advantages are

$$\mathsf{Exp}^{\mathsf{ind\text{-}cca}-b}_{\mathcal{E},\mathcal{A}}(\mathfrak{K})$$
1. $\mathsf{param} \leftarrow \mathsf{Setup}(1^{\mathfrak{K}})$
2. $(\mathsf{ek}, \mathsf{dk}) \leftarrow \mathsf{KeyGen}(\mathsf{param})$
3. $(\ell^*, m_0, m_1) \leftarrow \mathcal{A}(\texttt{FIND} : \mathsf{ek}, \mathsf{ODecrypt}(\cdot,\cdot))$
4. $c^* \leftarrow \mathsf{Encrypt}(\ell^*, \mathsf{ek}, m_b)$
5. $b' \leftarrow \mathcal{A}(\texttt{GUESS} : c^*, \mathsf{ODecrypt}(\cdot,\cdot))$
6. `IF` $(\ell^*, c^*) \in \mathcal{CT}$ `RETURN` 0
7. `ELSE RETURN` b'

$$\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\mathcal{A}) = \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}-1}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = 1] - \Pr[\mathsf{Exp}^{\mathsf{ind\text{-}cca}-0}_{\mathcal{E},\mathcal{A}}(\mathfrak{K}) = 1] \quad \mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(t) = \max_{\mathcal{A} \le t}\{\mathsf{Adv}^{\mathsf{ind\text{-}cca}}_{\mathcal{E}}(\mathcal{A})\}.$$

A.2 Statistical and Computational Distances

Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two probability distributions over a finite set $\mathcal{S}$ and let X and Y be two random variables with these two respective distributions.

Statistical Distance. The statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is also the statistical distance between X and Y:

$$\mathrm{Dist}(\mathcal{D}_1, \mathcal{D}_2) = \mathrm{Dist}(X, Y) = \sum_{x \in \mathcal{S}} |\Pr[X = x] - \Pr[Y = x]|.$$

If the statistical distance between $\mathcal{D}_1$ and $\mathcal{D}_2$ is less than or equal to ε, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are ε-close or are ε-statistically indistinguishable. If the $\mathcal{D}_1$ and D_2 are 0-close, we say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are perfectly indistinguishable.

Computational Distance. We say that $\mathcal{D}_1$ and $\mathcal{D}_2$ are (t, ε)-computationally indistinguishable, if, for every probabilistic algorithm $\mathcal{A}$ running in time at most t:

$$|\Pr[\mathcal{A}(X) = 1] - \Pr[\mathcal{A}(Y) = 1]| \le \varepsilon.$$

We can note that for any t and ε, $\mathcal{D}_1$ and $\mathcal{D}_2$ are (t, ε)-computationally indistinguishable, if they are ε-close.

A.3 Concrete Instantiations

All the analyses in this paper could be instantiated with ElGamal-like schemes, based on either the Decisional Diffie-Hellman (DDH) assumption, or the Decisional Linear (DLin) assumption. But we focus on the former only:

Definition 5 (Decisional Diffie-Hellman (DDH)). *The Decisional Diffie-Hellman assumption says that, in a group $(p, \mathbb{G}, g)$, when we are given (g^a, g^b, g^c) for unknown random $a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = ab \bmod p$ (a DH tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}^{\mathsf{ddh}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in distinguishing a DH tuple from a random tuple within time t.*

Cramer-Shoup (CS) Encryption Scheme [CS98]: it can be turned into a labeled public-key encryption scheme:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with a generator g
- $\mathsf{KeyGen}(\mathsf{param})$ generates $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$, $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and sets, $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h = g_1^z$. It also chooses a Collision-Resistant hash function $\mathfrak{H}_K$ in a hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r)$, for a message $M \in \mathbb{G}$ and a random scalar $r \in \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^{\xi})^r)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C)$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \stackrel{?}{=} v$. If the equality holds, one computes $M = e/u_1^z$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DDH assumption and the collision-resistance / universal one-wayness of the hash function $\mathcal{H}$.

B Security Proof for LAKE

B.1 Security Model

In this paper, we focus on efficiency and propose (in Section 5.1) an extension of the PAKE security model presented by Bellare-Pointcheval-Rogaway [BPR00] model for PAKE, between n players in the presence of an adversary. 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, described below:

- $\mathsf{Execute}(U, s, U', t)$: one outputs the transcript of an execution of the protocol between the instance Π_U^s of U and the instance $\Pi_{U'}^t$ of U'. It models passive eavesdropping attacks;
- $\mathsf{Send}(U, s, U', t, m)$: one sends the message m to the instance $\Pi_{U'}^t$ of U' in the name of the instance Π_U^s of U. It models active attacks;
- $\mathsf{Reveal}(U, s)$: if the instance Π_U^s of U has "accepted", one outputs the session key, otherwise one outputs $\perp$. It models a possible bad later use of the session key;
- $\mathsf{Test}(U, s)$: one first flips a coin $b \xleftarrow{\$} \{0, 1\}$, if $b = 1$ one outputs $\mathsf{Reveal}(U, s)$, otherwise one outputs a truly random key. It models the secrecy of the session key.

We say that Π_U^s and $\Pi_{U'}^t$ have matching conversations if inputs-outputs of the former correspond to the outputs-inputs of the latter and vice-versa. They are then called *partners*. We say that an instance is *fresh* if the key exists and is not trivially known by the adversary: more precisely, Π_U^s is fresh if

- Π_U^s has *accepted* the session, which is required to compute a session key;

- Π_U^s has not been asked a Reveal-query;
- no $\Pi_{U'}^t$ with *matching conversations* with Π_U^s has been asked a Reveal-query.

A key exchange protocol is then said secure if keys are indistinguishable from random keys for adversaries. Formally, the adversary is allowed to ask as many Execute, Send and Reveal-queries as it likes, and then only one Test-query to a *fresh* instance Π_U^s of a player. The adversary wins if it has guessed correctly the bit b in this query.

B.2 Proof of Theorem 4

This proof follows the one from [KV11]. It starts from the real attack game, in a Game 0: $\mathsf{Adv}_0(\mathcal{A}) = \varepsilon$. We incrementally modify the simulation to make possible the trivial attacks only. In the first games, all the honest players have their own values, and the simulator knows and can use them. Following [KV11], we can assume that there are two kinds of Send-queries: $\mathsf{Send}_0(U, s, U')$-queries where the adversary asks the instance Π_U^s to initiate an execution with an instance of U'. It is answered by the flow U' should send to communicate with U; $\mathsf{Send}_1(U, s, m)$-queries where the adversary sends the message m to the instance Π_U^s. It gives no answer back, but defines the session key, for possible later Reveal or Test-queries.

Game $\mathbf{G}_1$: We first modify the way Execute-queries are answered: we replace C and C' by encryptions of a fixed message M_0, that parses as two private parts P and P' and a word W, such that W is not in the language induced by (pub, P). Since the hashing keys are known, the common session key is computed as

$$\mathsf{sk} = \mathsf{Hash}(\mathsf{hk}, ((\mathsf{ek}, \mathsf{pub}), priv'), C') \times \mathsf{Hash}(\mathsf{hk}', ((\mathsf{ek}, \mathsf{pub}), \mathsf{priv}), C).$$

Since we could have first modified the way to compute sk, that has no impact at all from the soundness of the SPHF, the unique difference comes from the different ciphertexts. This is anyway indistinguishable under the IND-CPA property of the encryption scheme, for each Execute-query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_0(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_2$: We modify again the way Execute-queries are answered: we replace the common session key by a truly random value. Since the languages are not satisfied, the smoothness guarantees indistinguishability: $|\mathsf{Adv}_2(\mathcal{A}) - \mathsf{Adv}_1(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_3$: We now modify the way one answers the Send_1-queries, by using a decryption oracle, or alternatively knowing the decryption key. More precisely, when a message (hp, C) is sent, three cases can appear:

- it has been generated (altered) by the adversary, then one first decrypts the ciphertext to get $(\mathsf{priv}', priv, W')$ used by the adversary. Then
  - If they are correct ($W' \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'}$) and consistent with the receiver's values ($\mathsf{priv}' = priv', priv = \mathsf{priv}$) —event Ev— one declares that $\mathcal{A}$ succeeds (saying that $b' = b$) and terminates the game;
  - if they are not both correct and consistent with the receiver's values, one chooses sk at random.
- it is a replay of a previous flow sent by the simulator, then, in particular, one knows the hashing keys, and one can compute the session keys using all the hashing keys.

The first case can only increase the advantage of the adversary in case Ev happens (which probability is computed in $\mathbf{G}_6$). The second change is indistinguishable under the adaptive-smoothness and thus only increases the advantage of the adversary by a negligible term. The third change does not affect the way the key is computed, so finally: $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{negl}()$.

Game $\mathbf{G}_4$: We modify again the way one answers the Send_1-queries. More precisely, when a message (hp, C) is sent, two cases can appear:

- if there is an instance $\Pi_{U'}^t$ partnered with Π_U^s that receives this flow, then set the key identical to the key for $\Pi_{U'}^t$;
- otherwise, one chooses sk at random.

The former case remains identical since the message is a replay of a previous flow, and the latter is indistinguishable, as in [KV11], thanks to the adaptive-smoothness and their technical lemma that proves that all the hash values are random looking even when hashing keys and ciphertexts are re-used: $|\mathsf{Adv}_4(\mathcal{A}) - \mathsf{Adv}_3(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_5$: We now modify the way one answers the Send_0-queries: instead of encrypting the correct values, one does as in $\mathbf{G}_1$ for Execute-queries, and encrypts M_0. Since for simulating the Send_1-queries decryptions are required, indistinguishability relies on the IND-CCA security of the encryption scheme: $|\mathsf{Adv}_5(\mathcal{A}) - \mathsf{Adv}_4(\mathcal{A})| \leq \mathsf{negl}()$.

Game $\mathbf{G}_6$: For all the hashing and projection keys, we now use the dummy private inputs. Since we restricted hk and hp not to depend on aux, the distributions of these keys are independent of the auxiliary private inputs: $|\mathsf{Adv}_6(\mathcal{A}) - \mathsf{Adv}_5(\mathcal{A})| \leq \mathsf{negl}()$.

If one combines all the relations, one gets $\mathsf{Adv}_6(\mathcal{A}) \geq \mathsf{Adv}_0(\mathcal{A}) - \mathsf{negl}() = \varepsilon - \mathsf{negl}()$.

One can note that in this final game, the values of the honest players are not used anymore during the simulation, but just for declaring whether the adversary has won or not (event Ev). Otherwise, non-partnered players have random and independent keys, and thus unless the simulator stops the simulation, the advantage in the last game is exactly 0: $\mathsf{Adv}_6(\mathcal{A}) = \Pr[\mathsf{Ev}]$. And thus, we have $\varepsilon \leq \Pr[\mathsf{Ev}] + \mathsf{negl}()$.

Let us recall that Ev means that the adversary has encrypted $(\mathsf{priv}', priv, W')$ that are correct ($W' \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'}$) and consistent with the receiver's values ($\mathsf{priv}' = priv', priv = \mathsf{priv}$). Since the values for the honest players are never used during the simulation, we can assume we choose them at the very end only to check whether event Ev happened:

$$\Pr[\mathsf{Ev}] = \Pr[\exists k : \mathsf{priv}'(k) = priv'_{i_k}, priv(k) = priv_{i_k}, W'(k) \in \mathrm{L}_{\mathsf{pub},\mathsf{priv}'_{i_k}}]$$

where k lists all the Send_1-queries with adversary-generated messages in which the ciphertexts decrypt to $(\mathsf{priv}'(k), priv(k), W'(k))$, and i_k is the index of the recipient of k-th Send_1-query: it has first to guess the private values, and then once it has guessed them it has to find a word in the language:

$$\Pr[\mathsf{Ev}] \leq \frac{q_s}{2^m} \times \mathsf{Succ}^{\mathrm{L}}(t),$$

where m is the minimal min-entropy on the joint distributions of the $(\mathsf{priv}, priv')$ for any two players U, U' who want to communicate, and $\mathsf{Succ}^{\mathrm{L}}(t)$ is the best success an adversary can get in finding a word in a language $\mathrm{L}_{\mathsf{pub},\mathsf{priv}}$. Then, by combining all the inequalities, one gets

$$\varepsilon \leq \frac{q_s}{2^m} \times \mathsf{Succ}^{\mathrm{L}}(t) + \mathsf{negl}().$$

C Blind Signature

In this appendix, we give details on our two-flow Waters blind signature scheme outlined in Section 5.2. We first present the asymmetric variant of Waters signatures proposed in [BFPV11] and then recall the formal security definitions of blind signatures and of their security properties. Using the formalism from Appendix D, we describe in details the SPHF used in the scheme and finally prove the security of our scheme.

C.1 Waters Signature (Asymmetric Setting)

In 2011, Blazy, Fuchsbauer, Pointcheval and Vergnaud [BFPV11] proposed the following variant of Waters signatures in an asymmetric pairing-friendly environment:

- $\mathsf{Setup}(1^{\mathfrak{K}})$: in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$, one chooses a random vector $\boldsymbol{u} = (u_0, \ldots, u_\ell) \xleftarrow{\$} \mathbb{G}_1^{\ell+1}$, and for convenience, we denote $\mathcal{F}(M) = u_0 \prod_{i=1}^{\ell} u_i^{M_i}$ for $M = (M_i)_i \in \{0,1\}^\ell$. We also need two extra generators $(g_s, h_s) \xleftarrow{\$} \mathbb{G}_1^2$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e, g_s, h_s, \boldsymbol{u})$.
- $\mathsf{KeyGen}(\mathsf{param})$ chooses a random scalar $x \xleftarrow{\$} \mathbb{Z}_p$, which defines the public key as $\mathsf{vk} = (g_s^x, g_2^x) = (\mathsf{vk}_1, \mathsf{vk}_2)$, and the secret key is set as $\mathsf{sk} = h_s^x$.
- $\mathsf{Sign}(\mathsf{sk}, M; s)$ outputs, for some random $t \xleftarrow{\$} \mathbb{Z}_p$, $\sigma = \big(\sigma_1 = \mathsf{sk} \cdot \mathcal{F}(M)^t, \sigma_2 = (\sigma_{2,1} = g_s^t, \sigma_{2,2} = g_2^t)\big)$.
- $\mathsf{Verif}(\mathsf{vk}, M, \sigma)$ checks whether $e(\sigma_1, g_2) = e(h_s, \mathsf{vk}_2) \cdot e(\mathcal{F}(M), \sigma_{2,2})$, and $e(\sigma_{2,1}, g_2) = e(g_s, \sigma_{2,2})$.

This scheme is unforgeable against (adaptive) chosen-message attacks under the following variant of the CDH assumption, which states that CDH is hard in $\mathbb{G}_1$ when one of the random scalars is also given as an exponentiation in $\mathbb{G}_2$:

Definition 6 (The Advanced Computational Diffie-Hellman problem (CDH^+)). *In a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.*

C.2 Underlying SPHF in the Blind Signature Scheme

Following [BPV12], our scheme makes use of an SPHF in the interactive signing protocol to insure (in an efficient way) that the user actually knows the signed message. As outlined in Section 5.2, during the interactive process of the blind signature protocol, we have:

- General setting: a pairing-friendly system $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively;
- Encryption parameters: random scalars $(x_i)_i \in \mathbb{Z}_p^\ell$ with $(h_i = g_1^{x_i})_i$, where i ranges from 1 to ℓ, as everywhere in the following. Then, $\mathsf{ek} = (h_i)_i$;
- Signature parameters: independent generators $\boldsymbol{u} = (u_i)_{i \in \{0,\ldots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters function, $g_s = \prod_i h_i$, and a random generator $h_s \in \mathbb{G}_1$, then $\mathsf{sk} = h_s^x$ and $\mathsf{vk} = (g_s^x, g_2^x)$, for a random scalar x.

The user has generated $c_0 = g_1^r$ and $c_i = h_i^r u_i^{M_i}$ for $i = 1, \ldots, \ell$, as well as $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$. In the following simulation, we will extract $(M_i)_i$ from $C = (c_0, (c_i)_i)$, and we thus need to be sure that this message can be extracted. In addition, the simulator will also need to know vk_1^r to generate the blinded signature, hence its encryption in (d_0, d_1). But this has to be checked, with the following language membership, where we use notations from Appendix D:

1. each (c_0, c_i) encrypts a bit;

$$\Gamma(C) = \left(\begin{array}{c|c|c|c} g_1 & h_1 \ldots h_\ell & 1 \ldots 1 & 1 \ldots 1 \\ \hline 1 & u_1 \quad 1 & c_0 \quad 1 & c_1/u_1 \quad 1 \\ \vdots & \quad \ddots & \quad \ddots & \quad \ddots \\ 1 & 1 \quad u_\ell & 1 \quad c_0 & 1 \quad c_\ell/u_\ell \\ \hline 1 & 1 & g_1 \quad 1 & h_1 \quad 1 \\ \vdots & & \quad \ddots & \quad \ddots \\ 1 & & 1 \quad g_1 & 1 \quad h_\ell \end{array}\right) \qquad \begin{array}{l} \Theta_{\mathsf{aux}}(C) = (c_0, (c_i)_i, (1)_i, (1)_i) \\ \boldsymbol{\lambda} = (r, (M_i)_i, (-rM_i)_i) \end{array}$$

$$\boldsymbol{\lambda} \cdot \Gamma(C) = (g_1^r, (h_i^r u_i^{M_i})_i, (c_0^{M_i} g_1^{-rM_i})_i, ((c_i/u_i h_i^r)^{M_i})_i).$$

2. the ciphertext (d_0, d_1) encrypts the Diffie-Hellman value of $(g_1, c_0, \mathsf{vk}_1)$;

$$\Gamma = \begin{pmatrix} g_1 & 1 & \mathsf{vk}_1 \\ 1 & g_1 & h_1 \end{pmatrix} \qquad \begin{array}{l} \Theta_{\mathsf{aux}}(C) = (c_0, d_0, d_1) \qquad \boldsymbol{\lambda} = (r, s) \\ \boldsymbol{\lambda} \cdot \Gamma = (g_1^r, g_1^s, \mathsf{vk}_1^r h_1^s) \end{array}$$

The two matrices can be compressed with a common row/column: the same witness r is indeed used in both matrices, the two corresponding rows can be merged; the first column is the same in both matrices, it can thus be a common one:

$$\Gamma(C) = \left(\begin{array}{c|c|c|c|c|c} g_1 & h_1 \dots h_\ell & 1 \dots 1 & 1 & 1 \dots 1 & \mathsf{vk}_1 \\ \hline 1 & u_1 \quad 1 & c_0 \quad 1 & 1 & c_1/u_1 \quad 1 & 1 \\ \vdots & \ddots & \ddots & \vdots & \ddots & \vdots \\ 1 & 1 \quad u_\ell & 1 \quad c_0 & 1 & 1 \quad c_\ell/u_\ell & 1 \\ \hline 1 & & g_1 \quad 1 & 1 & h_1 \quad 1 & 1 \\ \vdots & 1 & \ddots & \vdots & \ddots & \vdots \\ 1 & & 1 \quad g_1 & 1 & 1 \quad h_\ell & 1 \\ \hline 1 & 1 \dots 1 & 1 \dots 1 & g_1 & 1 \dots 1 & h_1 \end{array}\right) \quad \begin{array}{l} \Theta_{\mathsf{aux}}(C) = (c_0, (c_i)_i, (1)_i, d_0, (1)_i, d_1) \\ \boldsymbol{\lambda} = (r, (M_i)_i, (-rM_i)_i, s) \end{array}$$

$$\boldsymbol{\lambda} \cdot \Gamma(C) = (g_1^r, (h_i^r u_i^{M_i})_i, (c_0^{M_i} g_1^{-rM_i})_i, g_1^s, ((c_i/u_i h_i^r)^{M_i})_i, \mathsf{vk}_1^r h_1^s).$$

This leads to, with $\mathsf{hk} = (\eta, \{\theta_i\}_i, \{\nu_i\}_i, \gamma, \{\mu_i\}_i, \lambda)$,

$$\mathsf{hp}_1 = g_1^\eta \cdot \prod_i h_i^{\theta_i} \cdot \mathsf{vk}_1^\lambda \qquad (\mathsf{hp}_{2,i} = u_i^{\theta_i} c_0^{\nu_i} (c_i/u_i)^{\mu_i})_i \qquad (\mathsf{hp}_{3,i} = g_1^{\nu_i} h_i^{\mu_i})_i \qquad \mathsf{hp}_4 = g_1^\gamma h_1^\lambda$$

$$H = c_0^\eta \cdot \prod_i c_i^{\theta_i} \cdot d_0^\gamma \cdot d_1^\lambda = \mathsf{hp}_1^r \cdot \prod_i \mathsf{hp}_{2,i}^{M_i} \cdot \mathsf{hp}_{3,i}^{-rM_i} \cdot \mathsf{hp}_4^s = H'.$$

The signers thus uses H to mask his blinded signature (σ_1', σ_2). But since σ_2 is just a random pair, only σ_1' needs to be masked. Without it, one cannot forge a signature, but it can be unmasked by the user with H', if the values $(c_0, (c_i)_i, (d_0, d_1))$ are in the correct language, and thus are correct ciphertexts.

One can note that the projection key consists of $2\ell + 2$ group elements in $\mathbb{G}_1$, and the hash value is in $\mathbb{G}_1$. No pairings are needed for this SPHF. Since Γ depends on C, this is a GL-SPHF, but this is enough for our interactive protocol.

C.3 Security proofs

Proposition 7. *Our blind signature scheme is blind under the* DDH *assumption in* $\mathbb{G}_1$[2]*:*

$$\mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(\mathfrak{K}) \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{DDH}}_{p,\mathbb{G}_1,g_1}(\mathfrak{K}).$$

Proof. Let us consider an adversary $\mathcal{A}$ against the blindness of our scheme. We build an adversary $\mathcal{B}$ against the DDH assumption in $\mathbb{G}_1$.

Game $\mathbf{G}_0$: In a first game $\mathbf{G}_0$, we run the standard protocol:

- $\mathsf{BSSetup}(1^k)$, $\mathcal{B}$ generates $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ with g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. It also generates independent generators $\boldsymbol{u} = (u_i)_{i \in \{0,\dots,\ell\}} \in \mathbb{G}_1^{\ell+1}$ for the Waters function and sets $\mathsf{ek} = (h_i)_i$ and $g_s = \prod_i h_i$. It generates $h_s = g_s^\alpha \in \mathbb{G}_1$ and defines the global parameters as $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \boldsymbol{u})$;
- The adversary $\mathcal{A}$ generates a verification key $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ such that $e(\mathsf{vk}_1, g_2) = e(g_s, \mathsf{vk}_2)$ and two ℓ-bit messages M^0, M^1.
- $\mathcal{A}$ and $\mathcal{B}$ run twice the interactive issuing protocol, first on the message M^b, and then on the message M^{1-b}:
  - $\mathcal{B}$ 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 $(c_i = h_i^r u_i^{M_i^b})_i$. $\mathcal{B}$ also encrypts vk_1^r, into $d_0 = g_1^s$, $d_1 = h_1^s \mathsf{vk}_1^r$ and sends $(c_0, (c_i)_i, (d_0, d_1))$ to $\mathcal{A}$.

[2] This assumption is sometimes referred to as the XDH assumption.

- $\mathcal{A}$ then outputs $(\mathsf{hp}, \Sigma = \sigma'_1 \times H, \sigma_2)$
- $\mathcal{B}$, using its witnesses and hp, computes H' and unmasks $\sigma'_1 = \Sigma/H$ which together with σ_2 should be a valid Waters Signature on M^b. It then randomizes the signature with s' to get Σ_b.

The same is done a second time with M^{1-b} to get Σ_{1-b}.

– $\mathcal{B}$ publishes (Σ_0, Σ_1).
– Eventually, $\mathcal{A}$ outputs b'.

We denote by ε the advantage of $\mathcal{A}$ in this game. By definition, we have:

$$\varepsilon = \mathsf{Adv}^{\mathsf{bl}}_{\mathcal{BS},\mathcal{A}}(k) = \Pr_{\mathbf{G}_0}[b' = 1|b = 1] - \Pr_{\mathbf{G}_0}[b' = 1|b = 0] = 2 \times \Pr_{\mathbf{G}_0}[b' = b] - 1.$$

Game $\mathbf{G}_1$: In a second game $\mathbf{G}_1$, we modify the way $\mathcal{B}$ extracts the signatures Σ_b and Σ_{1-b}. Since $\mathcal{B}$ knows the scalar α such that $h_s = g_s^\alpha$ it can compute the secret key $\mathsf{sk} = \mathsf{vk}_1^\alpha$ associated to $\mathsf{vk} = (\mathsf{vk}_1, \mathsf{vk}_2)$. One can note that, since we focus on valid executions with the signer, and due to the re-randomization of Waters signatures which leads to random signatures, $\mathcal{B}$ can generates itself random signatures on M^b and M^{1-b} using sk. This game is perfectly indistinguishable from the previous one:

$$\Pr_{\mathbf{G}_1}[b' = b] = \Pr_{\mathbf{G}_0}[b' = b].$$

Game $\mathbf{G}_2$: In this final game, we replace all the ciphertexts sent by $\mathcal{B}$ by encryption of random group elements in $\mathbb{G}_1$. For proving indistinguishability with the previous game, we use the hybrid technique for ElGamal ciphertexts with randomness re-use [BBS03]:

$$\varepsilon \leq 2 \times (\ell + 1) \times \mathsf{Adv}^{\mathsf{DDH}}_{p,\mathbb{G}_1,g_1}(\mathfrak{K}) + 2 \times \Pr_{\mathbf{G}_2}[b' = b] - 1.$$

In this last game, the two executions are thus perfectly indistinguishable, and thus $\Pr_{\mathbf{G}_2}[b' = b] = 1/2$ and we get the bound claimed in the proposition. □

Proposition 8. *Our blind signature scheme is unforgeable under the* CDH^+ *assumption.*

$$\mathsf{Adv}^{\mathsf{uf}}_{\mathcal{BS},\mathcal{A}}(\mathfrak{K}) \leq \Theta\left(\frac{\mathsf{Succ}^{\mathsf{CDH}^+}_{p,\mathbb{G}_1,g_1,\mathbb{G}_2,g_2}(\mathfrak{K})}{q_s\sqrt{\ell}}\right).$$

Proof. Let $\mathcal{A}$ be an adversary against the Unforgeability of the scheme. We assume that this adversary is able after q_s signing queries to output at least $q_s + 1$ valid signatures on different messages (for some q_s polynomial in the security parameter). We now build an adversary $\mathcal{B}$ against the CDH^+ assumption.

– $\mathcal{B}$ is first given a CDH^+ challenge $(g_s, g_2, g_s^x, g_2^x, h_s)$ in a pairing-friendly environment $(p, \mathbb{G}_1, g_1, \mathbb{G}_2, g_2, \mathbb{G}_T, e)$
– $\mathcal{B}$ emulates $\mathsf{BSSetup}$: it picks a random position $j \xleftarrow{\$} \{0, \ldots, \ell\}$, random indices $y_0, \ldots, y_\ell \xleftarrow{\$} \{0, \ldots, 2q_s - 1\}$ and random scalars $z_0, \ldots, z_\ell \xleftarrow{\$} \mathbb{Z}_p$ and publishes $\boldsymbol{u} = (u_i)_{i\in\{0,\ldots,\ell\}} \in \mathbb{G}^{\ell+1}$ for the Waters function, where $u_0 = h_s^{y_0 - 2jq_s} g_s^{z_0}$ and $u_i = h_s^{y_i} g_s^{z_i}$ for $i \in \{1, \ldots, \ell\}$. It sets $g_1 = g_s^\gamma$ and $\mathsf{ek} = (h_i)_i$ with $h_i = g_1^{a_i} \in \mathbb{G}_1$ for $i \in \{1, \ldots, \ell\}$ for some known random scalars $a_1, \ldots, a_\ell$ and $\gamma = 1/\sum_i a_i \bmod p$. It keeps secret the associated decryption key $\mathsf{dk} = (a_1, \ldots, a_\ell) \in \mathbb{Z}_p^\ell$ and outputs the global $\mathsf{param} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, \mathsf{ek}, g_s, h_s, \boldsymbol{u})$.
– $\mathcal{B}$ then emulates $\mathsf{BSKeyGen}$: it publishes $\mathsf{vk} = (g_s^x, g_2^x)$ from the challenge as its verification key (one can note that recovering the signing key h_s^x is the goal of our adversary $\mathcal{B}$);
– $\mathcal{A}$ can now interact q_s times with the signer, playing the interactive protocol $\mathsf{BSProtocol}\langle\mathcal{S}, \mathcal{A}\rangle$:
  - $\mathcal{A}$ sends the bit-per-bit encryptions c_i for $i \in \{1, \ldots, \ell\}$, and the extra ciphertext (d_0, d_1) hiding Y the verification key vk_1 raised to the randomness;

- Thanks to dk, $\mathcal{B}$ is able to extract M from the bit-per-bit ciphertexts (either the decryption leads to u_i and so $M_i = 1$, or to g_1 and so $M_i = 0$), and $Y = \mathsf{vk}_1^r$ from the additional ciphertext (d_0, d_1). One can also compute $c_0^{1/\gamma} = g_s^r$.
- If one of the extracted terms is not of the right form (either not a bit in the c_i, or $(g_s, g_s^r, \mathsf{vk}_1, Y)$ is not a Diffie-Hellman tuple, which occurs if $e(g_s^r, \mathsf{vk}_2) \neq e(Y, g_2)$ and can thus be checked with a pairing computation), then $\mathcal{A}$ has submitted a "word" not in the appropriate language for the SPHF. Therefore through the smoothness property of the SPHF, it is impossible from a theoretic point of view that the adversary extracts anything from $\mathcal{B}$'s answer, therefore $\mathcal{B}$ simply sends a random element Σ in $\mathbb{G}_1$ together with a valid random pair (g_1^t, g_2^t).
- If $(g_s, g_s^r, \mathsf{vk}_1, Y)$ is a Diffie-Hellman tuple, one knows that $Y = \mathsf{vk}_1^r$.
  $\mathcal{B}$ computes $H = -2jq_s + y_0 + \sum y_i M_i$ and $J = z_0 + \sum z_i M_i$, $\mathcal{F}(M) = h_s^H g_s^J$. If $H \equiv 0 \mod p$, it aborts, else it sets

$$\sigma = (\mathsf{vk}_1^{-J/H} Y^{-1/H} (\mathcal{F}(M) c_0^{1/\gamma})^s, (\mathsf{vk}_1^{-1/H} g_1^s, \mathsf{vk}_2^{-1/H} g_2^s)),$$

  for some random scalar s. Setting $t = s - x/H$, we can see this is indeed a valid signature (as output as the end of the signing interactive protocol), since we have:

$$\begin{aligned}
\sigma_1 &= \mathsf{vk}_1^{-J/H} Y^{-1/H} (\mathcal{F}(M) c_0^{1/\gamma})^s = \mathsf{vk}_1^{-J/H} g_s^{-xr/H} (h_s^H g_s^J g_s^r)^s \\
&= g_s^{-xJ/H} g_s^{-xr/H} (h_s^H g_s^J g_s^r)^t (h_s^H g_s^J g_s^r)^{x/H} = h^x (h^H g_s^J g_s^r)^t \\
&= \mathsf{sk} \cdot \delta^t \qquad \text{where } \delta = \mathcal{F}(M) \times g_s^r \\
\sigma_{2,1} &= \mathsf{vk}_1^{-1/H} g_1^s = g_1^{-x/H} g_1^s = g_1^t \\
\sigma_{2,2} &= \mathsf{vk}_2^{-1/H} g_2^s = g_2^{-x/H} g_2^s = g_2^t
\end{aligned}$$

- $\mathcal{B}$ then acts honestly to send the signature through the SPHF.

After a q_s queries, $\mathcal{A}$ outputs a valid signature σ^* on a new message M^* with non negligible probability.

– As before $\mathcal{B}$ computes $H^* = -2jq_s + y_0 + \sum y_i M_i^*$ and $J^* = z_0 + \sum z_i M_i^*$, $\mathcal{F}(M) = h^{H^*} g_1^{J^*}$.
– If $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ aborts. Otherwise $\sigma^* = (\mathsf{sk} \cdot \mathcal{F}(M^*)^t, g_s^t, g_2^t) = (\mathsf{sk} \cdot g_s^{tJ^*}, g_s^t, g_2^t)$ and so $\sigma_1^*/\sigma_2^{*J^*} = \mathsf{sk} = h_s^x$. Therefore if $\mathcal{A}$'s signature is valid and if $H^* \not\equiv 0 \mod p$, $\mathcal{B}$ solves its CDH^+ challenge.

The probability that all the $H \not\equiv 0 \mod p$ for all the simulations, but $H^* \equiv 0 \mod p$ in the forgery is the $(1, q_s)$-programmability of the Waters function. A full proof showing that it happens with probability in $\Theta(\mathsf{Succ}^{\mathsf{CDH}}_{p,\mathbb{G}_1,g_1,\mathbb{G}_2,g_2}(\mathfrak{K})/q_s\sqrt{\ell})$ can be found in [HK08]. □

D Generic Framework for SPHFs and New Constructions

In this appendix, we introduce our full generic framework for SPHFs using a new notion of graded rings, derived from [GGH12]. It enables to deal with cyclic groups, bilinear groups (with symmetric or asymmetric pairings), or even groups with multi-linear maps. Namely, it handles all the previous constructions from [BBC+13].

Before introducing graded rings and our generic framework, we briefly recall the definition of bilinear groups. The last three subsections are dedicated to instantiations. The last instantiation can deal with any quadratic pairing product equation over ciphertexts, which encompasse all languages handled by Groth-Sahai NIZKs, and so can deal with any NP language. We can see that our generic scheme greatly simplify the construction and the presentation of all the SPHFs presented in these last subsections.

This appendix is very formal and technical. We strongly recommend the reader to first read Sections D.3 and 4 where we give the intuition.

D.1 Bilinear Groups

Let us consider three multiplicative cyclic groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of prime order p. Let g_1 and g_2 be two generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ or $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is called a *bilinear setting* if $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ is a bilinear map (called a pairing) with the following properties:

- *Bilinearity.* For all $(a, b) \in \mathbb{Z}_p^2$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$;
- *Non-degeneracy.* The element $e(g_1, g_2)$ generates $\mathbb{G}_T$;
- *Efficient computability.* The function e is efficiently computable.

It is called a *symmetric* bilinear setting if $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$. In this case, we denote it $(p, \mathbb{G}, \mathbb{G}_T, e)$ and we suppose $g = g_1 = g_2$. Otherwise, if $\mathbb{G}_1 \neq \mathbb{G}_2$, it is called an *asymmetric* bilinear setting one otherwise.

D.2 Graded Rings

Our graded rings are a practical way to manipulate elements of various groups involved with pairings, and more generally, with multi-linear maps. This is a slight variant of the notion of graded encoding proposed in [GGH12], where each element has only one representation, instead of a set of representations, and where we can add two elements even with different indexes.

Indexes Set. As in [GGH12], let us consider a finite set of indexes $\Lambda = \{0, \ldots, \kappa\}^\tau \subset \mathbb{N}^\tau$. In addition to considering the addition law $+$ over Λ, we also consider Λ as a bounded lattice, with the two following laws:

$$\sup(\boldsymbol{v}, \boldsymbol{v}') = (\max(\boldsymbol{v}_1, \boldsymbol{v}'_1), \ldots, \max(\boldsymbol{v}_\tau, \boldsymbol{v}'_\tau)) \qquad \inf(\boldsymbol{v}, \boldsymbol{v}') = (\min(\boldsymbol{v}_1, \boldsymbol{v}'_1), \ldots, \min(\boldsymbol{v}_\tau, \boldsymbol{v}'_\tau)).$$

We also write $\boldsymbol{v} < \boldsymbol{v}'$ (resp. $\boldsymbol{v} \leq \boldsymbol{v}'$) if and only if for all $i \in \{1, \ldots, \tau\}$, $\boldsymbol{v}_i < \boldsymbol{v}'_i$ (resp. $\boldsymbol{v}_i \leq \boldsymbol{v}'_i$). Let $\bar{0} = (0, \ldots, 0)$ and $\top = (\kappa, \ldots, \kappa)$, be the minimal and maximal elements.

Graded Ring. The (κ, τ)-graded ring for a commutative ring R is the set $\mathfrak{G} = \Lambda \times R = \{[\boldsymbol{v}, x] \,|\, \boldsymbol{v} \in \Lambda, x \in R\}$, where $\Lambda = \{0, \ldots, \kappa\}^\tau$, with two binary operations $(+, \cdot)$ defined as follows:

- for every $u_1 = [\boldsymbol{v}_1, x_1], u_2 = [\boldsymbol{v}_2, x_2] \in \mathfrak{G}$: $u_1 + u_2 \stackrel{\text{def}}{=} [\sup(\boldsymbol{v}_1, \boldsymbol{v}_2), x_1 + x_2]$;
- for every $u_1 = [\boldsymbol{v}_1, x_1], u_2 = [\boldsymbol{v}_2, x_2] \in \mathfrak{G}$: $u_1 \cdot u_2 \stackrel{\text{def}}{=} [\boldsymbol{v}_1 + \boldsymbol{v}_2, x_1 \cdot x_2]$ if $\boldsymbol{v}_1 + \boldsymbol{v}_2 \in \Lambda$, or $\perp$ otherwise, where $\perp$ means the operation is undefined and cannot be done.

We remark that $\cdot$ is only a partial binary operation and we use the following convention: $\perp + u = u + \perp = u \cdot \perp = \perp \cdot u = \perp$, for any $u \in \mathfrak{G} \cup \{\perp\}$. We then denote $\mathfrak{G}_{\boldsymbol{v}}$ the additive group $\{u = [\boldsymbol{v}', x] \in \mathfrak{G} \,|\, \boldsymbol{v}' = \boldsymbol{v}\}$. We will make natural use of vector and matrix operations over graded ring elements.

Cyclic Groups and Pairing-Friendly Settings. In the sequel, we consider graded rings over $R = \mathbb{Z}_p$ only, because we will use the vectorial space structure over $\mathbb{Z}_p$ in the proof of the smoothness of our generic construction of SPHF (see Section D.3). This means we cannot directly deal with constructions in [GGH12] yet. Nevertheless, graded rings enable to easily deal with cyclic groups $\mathbb{G}$ of prime order p, and bilinear groups.

Cyclic Group In this case, $\kappa = \tau = 1$: elements $[0, x]$ of index 0 correspond to scalars $x \in \mathbb{Z}_p$ and elements $[1, x]$ of index 1 correspond to group elements $g^x \in \mathbb{G}$.

Symmetric Bilinear Group. Let $(p, \mathbb{G}, \mathbb{G}_T, e)$ be a symmetric bilinear group, and g be a generator of $\mathbb{G}$. We can represent this bilinear group by a graded ring $\mathfrak{G}$ with $\kappa = 2$ and $\tau = 1$. More precisely, we can consider the following map: $[0, x]$ corresponds to $x \in \mathbb{Z}_p$, $[1, x]$ corresponds to $g^x \in \mathbb{G}$ and $[2, x]$ corresponds to $e(g, g)^x \in \mathbb{G}_T$.

Asymmetric Bilinear Group. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be an asymmetric bilinear group, and g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. We can represent this bilinear group by a graded ring $\mathfrak{G}$ with $\kappa = 1$ and $\tau = 2$. More precisely, we can consider the following map: $[(0,0),x]$ corresponds to $x \in \mathbb{Z}_p$, $[(1,0),x]$ corresponds to $g_1^x \in \mathbb{G}_1$, $[(0,1),x]$ corresponds to $g_2^x \in \mathbb{G}_2$ and $[(1,1),x]$ corresponds to $e(g_1,g_2)^x \in \mathbb{G}_T$.

Notations. We have chosen an additive notation for the group law in $\mathfrak{G}_{\boldsymbol{v}}$. On the one hand, this a lot easier to write generic things done, but, on the other hand, it is a bit cumbersome for bilinear groups to use additive notations. Therefore, when we provide an example with a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, we use multiplicative notation $\cdot$ for the law in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, and additive notation $+$ for the law in $\mathbb{Z}_p$, as soon as it is not too complicated. But when needed, we will also use the notation $\oplus$ and $\odot$ which correspond to the addition law and the multiplicative law of the corresponding graded rings. In other words, for any $x, y \in \mathbb{Z}_p$, $u_1, v_1 \in \mathbb{G}_1$, $u_2, v_2 \in \mathbb{G}_2$ and $u_T, v_T \in \mathbb{G}_T$, we have:

$$\begin{array}{rcr} x \oplus y = x + y & & x \odot y = x \cdot y = xy \\ u_1 \oplus v_1 = u_1 \cdot v_1 = u_1 v_1 & & x \odot u_1 = u_1^x \\ u_2 \oplus v_2 = u_2 \cdot v_2 = u_2 v_2 & & x \odot u_1 = u_1^x \\ u_T \oplus v_T = u_T \cdot v_T & u_1 \odot u_2 = e(u_1,u_2) & x \odot u_T = u_T^x. \end{array}$$

The element 1 will always denote the neutral element in either $\mathbb{G}_1$, $\mathbb{G}_2$ or $\mathbb{G}_T$ (depending on the context) and not $1 \in \mathbb{Z}_p$, which is not used in our constructions.

D.3 Generic Framework for GL-SPHF/KV-SPHF

In this section, we exhibit a generic framework for SPHF for languages of ciphertexts. This is an extension of the framework described in Section 3 to graded rings. We assume that crs is fixed and we write $\mathrm{L}_{\mathsf{aux}} = \mathrm{LoFC}_{\mathsf{full\text{-}aux}} \subseteq \mathcal{S}et$ 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 $\Gamma : \mathcal{S}et \mapsto \mathfrak{G}^{k\times n}$, and a family of functions $\Theta_{\mathsf{aux}} : \mathcal{S}et \mapsto \mathfrak{G}^{1\times n}$, such that for any word $C \in \mathcal{S}et$, $(C \in \mathrm{L}_{\mathsf{aux}}) \iff (\exists \boldsymbol{\lambda} \in \mathfrak{G}^{1\times k}$ such that $\Theta_{\mathsf{aux}}(C) = \boldsymbol{\lambda} \cdot \Gamma(C))$. If Γ is a constant function (independent of the word C), this defines a KV-SPHF, otherwise this is a GL-SPHF. However, in any case, we need the indexes of the components of $\Gamma(C)$ to be independent of C.

We furthermore require that a user, who knows a witness w of the membership $C \in \mathrm{L}_{\mathsf{aux}}$, can efficiently compute $\boldsymbol{\lambda}$.

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, $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \cdot \boldsymbol{\alpha} \in \mathfrak{G}^k$ (if Γ does not depend on C, hp does not depend on C either). Then, the hash value is:

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

The set Π of hash values is exactly $\mathfrak{G}_{\boldsymbol{v}_H}$, the set of graded elements of index $\boldsymbol{v}_H$, the maximal index of the elements of $\Theta_{\mathsf{aux}}(C)$.

In addition, the following security analysis proves that the above generic SPHF is perfectly smooth, and thus proves the Theorem 2 as a particular case. We insist that if Γ really depends on C this construction yields a GL-SPHF, whereas when Γ is a constant matrix, we obtain a KV-SPHF, but perfectly smooth in both cases.

Security Analysis. In order to prove the smoothness of the above SPHF, we consider a word $C \notin \mathrm{L}_{\mathsf{aux}}$ and a projection key $\mathsf{hp} = \boldsymbol{\gamma}(C) = \Gamma(C) \cdot \boldsymbol{\alpha}$: $\forall \boldsymbol{\lambda} \in \mathfrak{G}^{1\times k}, \Theta_{\mathsf{aux}}(C) \neq \boldsymbol{\lambda} \cdot \Gamma(C)$. Using the projection $\mathfrak{L} : \mathfrak{G} \to \mathbb{Z}_p; u = [\boldsymbol{v}, x] \mapsto x$, which can be seen as the discrete logarithm, and which can be applied component-wise on vectors and matrices, this means that $\mathfrak{L}(\Theta_{\mathsf{aux}}(C))$ is linearly independent from the rows of $\mathfrak{L}(\Gamma(C))$. As a consequence, since $\boldsymbol{\alpha}$ is uniformly random, $\mathfrak{L}(\Theta_{\mathsf{aux}}(C)) \cdot \boldsymbol{\alpha}$ is a random variable independent from $\mathfrak{L}(\boldsymbol{\gamma}(C)) = \mathfrak{L}(\Gamma(C)) \cdot \boldsymbol{\alpha}$, and so from $\mathsf{hp} = \boldsymbol{\gamma}(C)$, since the index of $\boldsymbol{\gamma}(C)$ is a constant and thus $\mathfrak{L}(\boldsymbol{\gamma}(C))$ completely defines $\boldsymbol{\gamma}(C)$. Therefore, H is a uniform element of $\mathfrak{G}_{\boldsymbol{v}_H}$ given hp, aux and C.

D.4 Instantiations

A First Example with Pairings.

Notations. We consider the same kind of equation as in the body of the paper (Section 4.1), but on possibly two different groups $\mathbb{G}_1$ and $\mathbb{G}_2$, of the same prime order p, generated by g_1 and g_2, respectively, with a possible bilinear map into $\mathbb{G}_T$. We assume the DDH assumption hold in both $\mathbb{G}_1$ and $\mathbb{G}_2$. We define ElGamal encryption schemes with encryption keys $\mathsf{ek}_1 = (g_1, h_1 = g_1^{x_1})$ and $\mathsf{ek}_2 = (g_2, h_2 = g_2^{x_2})$ on each group. We are interested in languages on the ciphertexts $C_{1,i} = (u_{1,i} = g_1^{r_{1,i}}, e_{1,i} = h_1^{r_{1,i}} \cdot X_i)$, for $X_1, \ldots, X_{n_1} \in \mathbb{G}_1$, and $C_{2,j} = (u_{2,j} = g_2^{r_{2,j}}, e_{2,j} = h_2^{r_{2,j}} \cdot g_2^{y_j})$, for $y_1, \ldots, y_{n_2} \in \mathbb{Z}_p$, such that:

$$\prod_{i=1}^{n_1} X_i^{a_i} \cdot \prod_{j=1}^{n_2} A_j^{y_j} = B, \text{ with } \begin{array}{l}\mathsf{crs} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathsf{ek}_1, \mathsf{ek}_2)\\ \mathsf{aux} = (a_1, \ldots, a_{n_1}, A_1, \ldots, A_{n_2}, B) \in \mathbb{Z}_p^{n_2} \times \mathbb{G}_1^{n_2+1}.\end{array} \tag{2}$$

We insist that here, contrarily to equation (1) in Section 4.1, the group elements $(A_1, \ldots, A_{n_2})$ are part of aux, and thus not known in advance. The matrix Γ cannot depend on them anymore:

$$\Gamma = \left(\begin{array}{c|ccc|c} g_1 & 1 & \ldots & 1 & h_1 \\ \hline 1 & g_2 & & 1 & h_2 \\ \vdots & & \ddots & & \vdots \\ 1 & 1 & & g_2 & h_2 \end{array}\right) \qquad \begin{array}{rl} \Theta_{\mathsf{aux}}(\boldsymbol{C}) = & \left(\prod_i u_{1,i}^{a_i}, (e(A_j, u_{2,j}))_j, \prod_i e(e_{1,i}^{a_i}, g_2) \cdot \prod_j e(A_j, e_{2,j})/e(B, g_2)\right) \\ \boldsymbol{\lambda} = & (\sum_i a_i r_{1,i}, (A_j^{r_{2,j}})_j) \\ \boldsymbol{\lambda} \cdot \Gamma = & \left(g_1^{\sum_i a_i r_{1,i}}, (e(A_j, g_2^{r_{2,j}}))_j, e(h_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_{2,j}}, h_2)\right) \end{array}$$

We recall that in the matrix, 0 means $[\boldsymbol{v}, 0]$ for the appropriate index $\boldsymbol{v}$, and thus $1_{\mathbb{G}_1} = g_1^0 \in \mathbb{G}_1$ in the first line and column, but $1_{\mathbb{G}_2} = g_2^0 \in \mathbb{G}_2$ in the diagonal block. In addition, in the product $\boldsymbol{\lambda} \cdot \Gamma$, when adding two elements, they are first lifted in the minimal common higher ring, and when multiplying two elements, we either make a simple exponentiation (scalar with a group element) or a pairing (two group elements from different groups).

Because of the diagonal blocks in Γ, $\boldsymbol{\lambda}$ is implied by all but last components of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$, then the last column defines the relation: the last component of $\Theta_{\mathsf{aux}}(\boldsymbol{C})$ is $\prod_i e(h_1^{r_{1,i}a_i} X^{a_i}, g_2) \cdot \prod_j e(A_j, h_2^{r_{2,j}} g_2^{y_j})/e(B, g_2)$, which is equal to the last component of $\boldsymbol{\lambda} \cdot \Gamma$, multiplied by the expression below, that is equal to 1 if and only if the relation (2) is satisfied:

$$\prod\nolimits_i e(X^{a_i}, g_2) \cdot \prod\nolimits_j e(A_j, g_2^{y_j})/e(B, g_2) = e\left(\prod\nolimits_i X^{a_i} \cdot \prod\nolimits_j A_j^{y_j}/B, g_2\right).$$

It thus leads to the following KV-SPHF, with $\mathsf{hp}_1 = g_1^{\nu} h_1^{\lambda}$ and $(\mathsf{hp}_{2,j} = g_2^{\theta_j} h_2^{\lambda})_j$, for $\mathsf{hk} = (\nu, (\theta_j)_j, \lambda)$:

$$H = \prod\nolimits_i e((u_{1,i}^{\nu} e_{1,i}^{\lambda})^{a_i}, g_2) \cdot \prod\nolimits_j e(A_j, u_{2,j}^{\theta_j} e_{2,j}^{\lambda}) \cdot e(B^{-\lambda}, g_2) = e(\mathsf{hp}_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod\nolimits_j e(A_j^{r_{2,j}}, \mathsf{hp}_{2,j}) = H'.$$

As a consequence, the ciphertexts and the projection keys (which have to be exchanged in a protocol) globally consist of $2n_1 + 1$ elements from $\mathbb{G}_1$ and $3n_2$ elements from $\mathbb{G}_2$, and pairings are required for the hash value.

Ciphertexts with Randomness Reuse. We can apply the same improvement as in Section 4.1 by using multiple independent encryption keys in $\mathbb{G}_2$, $\mathsf{ek}_{2,j} = (g_2, h_{2,j} = g_2^{x_{2,j}})$, for $j = 1, \ldots, n_2$. This allows to reuse the same random coins [BBS03]. We are interested in languages on the ciphertexts $(C_{1,i} = (u_{1,i} = g_1^{r_{1,i}}, e_{1,i} = h_1^{r_{1,i}} \cdot X_i))_i$, for $(X_i)_i \in \mathbb{G}_1^{n_1}$, with $(r_{1,i})_i \in \mathbb{Z}_p^{n_1}$, and $C_2 = (u_2 = g_2^{r_2}, (e_{2,j} = h_{2,j}^{r_2} \cdot g_2^{y_j})_j)$, for $(y_j)_j \in \mathbb{Z}_p^{n_2}$, with $r_2 \in \mathbb{Z}_p$, still satisfying the same relation (2). This improves on the length of the ciphertexts of the g^{y_i}'s, from $2n_2$ group elements in $\mathbb{G}_2$ to $n_2 + 1$ in $\mathbb{G}_2$. A similar KV-SPHF as before can be derived, just modifying the last column vector $(h_2)_j$ by $(h_{2,j})_j$:

$$\Gamma = \left(\begin{array}{c|c|c} g_1 & 1 \ldots 1 & h_1 \\ \hline 1 & g_2 \quad 1 & h_{2,1} \\ \vdots & \quad \ddots & \vdots \\ 1 & 1 \quad g_2 & h_{2,n_2} \end{array}\right) \qquad \begin{array}{rl} \Theta_{\mathsf{aux}}(\boldsymbol{C}) = & \left(\prod_i u_{1,i}^{a_i}, (e(A_j, u_{2,j}))_j, \prod_i e(e_{1,i}^{a_i}, g_2) \cdot \prod_j e(A_j, e_{2,j})/e(B, g_2)\right) \\ \boldsymbol{\lambda} = & (\sum_i a_i r_{1,i}, (A_j^{r_2})_j) \\ \boldsymbol{\lambda} \cdot \Gamma = & \left(g_1^{\sum_i a_i r_{1,i}}, (e(A_j, g_2^{r_2}))_j, e(h_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_2}, h_{2,j})\right) \end{array}$$

It leads to the following KV-SPHF, with $\mathsf{hp}_1 = g_1^\nu h_1^\lambda$ and $(\mathsf{hp}_{2,j} = g_2^{\theta_j} h_{2,j}^\lambda)_j$, for $\mathsf{hk} = (\nu, (\theta_j)_j, \lambda)$:

$$H = \prod_i e((u_{1,i}^\nu e_{1,i}^\lambda)^{a_i}, g_2) \cdot \prod_j e(A_j, u_{2,j}^{\theta_j} e_{2,j}^\lambda) \cdot e(B^{-\lambda}, g_2) = e(\mathsf{hp}_1^{\sum_i a_i r_{1,i}}, g_2) \cdot \prod_j e(A_j^{r_2}, \mathsf{hp}_{2,j}) = H'.$$

Globally, the ciphertexts and the projection keys consist of $2n_1 + 1$ elements from $\mathbb{G}_1$ and $2n_2 + 1$ elements from $\mathbb{G}_2$, but pairings are still required for the hash value. The prior knowledge of the A_j's allows to avoid pairings, as shown in Section 4.1.

SPHF for Linear Pairing Equations over Ciphertexts. Let us now construct an KV-SPHF for a linear pairing equation in an asymmetric bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2)$ over ElGamal commitments. This will actually be a particular case of the construction of the next section for quadratic pairing equation. It is thus a warm-up for this more technical instantiation. The construction can obviously be extended to systems of linear pairing equations, and to other commitments schemes using the same methods as in Section 4. It can also be slightly simplified in the case of symmetric bilinear groups.

Notations. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be a (asymmetric) bilinear group. Let g_1, g_2 be generators of $\mathbb{G}_1, \mathbb{G}_2$ respectively, and let $g_T = e(g_1, g_2)$. Let $\mathsf{ek}_1 = (g_1, h_1 = g_1^{x_1})$, $\mathsf{ek}_2 = (g_2, h_2 = g_2^{x_2})$ and $\mathsf{ek}_T = (g_T, h_T = g_T^{x_T})$ be ElGamal key for encryption scheme in, respectively, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.

We are interested in languages of commitments $(C_{1,i})_i$ of $(X_{1,i})_i \in \mathbb{G}_1^{n_1}$, $(C_{2,j})_j$ of $(X_{2,j})_j \in \mathbb{G}_2^{n_2}$, and $(C_{T,k})_i$ of $(X_{T,k})_k \in \mathbb{G}_T^{n_T}$ such that:

$$\prod_i e(X_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, X_{2,j}) \cdot \prod_k X_{T,k}^{a_{T,k}} = B, \tag{3}$$

with $\mathsf{aux} = ((A_{1,j})_j, (A_{2,i})_i, (a_{T,k})_k) \in \mathbb{G}_1^{n_2} \times \mathbb{G}_2^{n_1} \times \mathbb{Z}_p^{n_T}$. This can also be written:

$$\left(\bigoplus_{i=1}^{n_1} A_{2,i} \odot X_{1,i}\right) \oplus \left(\bigoplus_{j=1}^{n_2} A_{1,j} \odot X_{2,j}\right) \oplus \left(\bigoplus_{k=1}^{n_T} a_{T,k} \odot X_{T,k}\right) = B.$$

Let us also write, for any $\omega \in \{1, 2, T\}$ and $\iota \in \{1, \ldots, n_\omega\}$: $C_{\omega,\iota} = (u_{\omega,\iota} = g_\omega^{r_{\omega,\iota}}, e_{\omega,\iota} = h_\omega^{r_{\omega,\iota}} X_{\omega,\iota})$. Words of $\mathcal{S}et$ are tuple $C = (C_{\omega,\iota})_{\omega \in \{1,2,T\}, \iota \in \{1,\ldots,n_\omega\}}$.

Basic Scheme in $\mathbb{G}_T$. Let us consider

$$\Gamma = \begin{pmatrix} g_1 & 1 & 1 & h_1 \\ 1 & g_2 & 1 & h_2 \\ 1 & 1 & g_T & h_T \end{pmatrix} \qquad \Theta(C) = \begin{pmatrix} \bigoplus_i A_{2,i} \odot u_{1,i}, \bigoplus_j A_{1,j} \odot u_{2,j}, \bigoplus_k a_{T,k} \odot u_{T,k}, \\ \qquad (\bigoplus_i A_{2,i} \odot e_{1,i}) \oplus \left(\bigoplus_j A_{1,j} \odot e_{2,j}\right) \oplus (\bigoplus_k a_{T,k} \odot e_{T,k}) \ominus B \end{pmatrix}.$$

Because of the diagonal block in Γ, one can note that the unique possibility is

$$\boldsymbol{\lambda} = (\bigoplus_i A_{2,i} \odot r_{1,i}, \bigoplus_j A_{1,j} \odot r_{2,j}, \bigoplus_k r_{T,k}) = (\prod_i A_{2,i}^{r_{1,i}}, \prod_j A_{1,j}^{r_{2,j}}, \sum_k r_{T,k}).$$

We then have $\boldsymbol{\lambda} \odot \Gamma = \Theta(C)$ if and only if

$$\prod_i e(h_1^{r_{1,i}}, A_{2,i}) \cdot \prod_j e(A_{1,j}, h_2^{r_{2,j}}) \cdot \prod_k h_T^{r_{T,k}} = \prod_i e(e_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, e_{2,j}) \cdot \prod_k e_{T,k}^{a_{T,k}} / B$$

and thus if and only if Equation (3) is true, *i.e.*, the word is in the language. Furthermore, if we set $\gamma_1 = g_1^{\alpha_1} h_1^{\alpha_4}$, $\gamma_2 = g_2^{\alpha_2} h_2^{\alpha_4}$, and $\gamma_3 = g_T^{\alpha_3} h_T^{\alpha_4}$, we have

$$\begin{aligned} H &= \left(\prod_{i=1}^{n_1} e(u_{1,i}, A_{2,i})\right)^{\alpha_1} \cdot \left(\prod_{j=1}^{n_2} e(A_{1,j}, u_{2,j})\right)^{\alpha_2} \cdot \left(\prod_{k=1}^{n_T} u_{T,k}^{a_{T,k}}\right)^{\alpha_3} \\ &\times \left(\prod_{i=1}^{n_1} e(e_{1,i}, A_{2,i}) \cdot \prod_{j=1}^{n_2} e(A_{1,j}, e_{2,j}) \cdot \prod_{k=1}^{n_T} e_{T,k}^{a_{T,k}} / B\right)^{\alpha_4} \\ &= e(\gamma_1, \prod_i A_{2,i}^{r_{1,i}}) \cdot e(\prod_j A_{1,j}^{r_{2,j}}, \gamma_2) \cdot \gamma_3^{\sum_k r_{T,k}} = H'. \end{aligned}$$

Variant. The above scheme is not efficient enough for practical use because elements in $\mathbb{G}_T$ are often big and operations in $\mathbb{G}_T$ are often slow. If $h_T = e(h_1, g_2)$, then the last row of Γ can be $(0, 0, g_1, h_1)$ which enables faster hashing and shorter projection key. We remark this modified encryption scheme in $\mathbb{G}_T$ is IND-CPA as soon as DDH is hard in $\mathbb{G}_1$, which we need to suppose for the ElGamal encryption scheme in $\mathbb{G}_1$ to be IND-CPA. So this variant is always more efficient when using ElGamal encryption.

However, if DDH is easy, as in symmetric bilinear group, this variant may not be interesting, since it requires to use the linear encryption scheme in $\mathbb{G}_T$ instead of the ElGamal one.

SPHF for Quadratic Pairing Equations over Ciphertexts. In this section, we present a KV-SPHF for language of ElGamal commitments verifying a quadratic pairing equation. As usual, it can be extended to systems of quadratic pairing equations, and to other commitments schemes. We use the same notations as in the previous construction.

Example. Before showing the generic construction, we describe it on a simple example: we are interested in languages of the ciphertexts $C_1 = (u_1 = g_1^{r_1}, e_1 = h_1^{r_1} X_1)$ and $C_2 = (u_2 = g_2^{r_2}, e_2 = h_2^{r_2} X_2)$, that encrypt two values X_1 and X_2 such that $e(X_1, X_2) = B$ where B is some constant in $\mathbb{G}_T$ and $\mathsf{aux} = B$. We remark the equation $e(X_1, X_2) = B$ can also be written $X_1 \odot X_2 = B$. Let us consider

$$\Gamma = \begin{pmatrix} g_1 \odot g_2 & 1 & 1 & h_1 \odot h_2 \\ 1 & g_1 & 1 & h_1 \\ 1 & 1 & g_2 & h_2 \end{pmatrix} \qquad \begin{aligned} \Theta(C) &= (-u_1 \odot u_2, u_1 \odot e_2, e_1 \odot u_2, e_1 \odot e_2 \ominus B) \\ &= (e(u_1, u_2)^{-1}, e(u_1, e_2), e(e_1, u_2), e(e_1, e_2)/B). \end{aligned}$$

Because of the diagonal block in Γ, one can note that the unique possibility is

$$\boldsymbol{\lambda} = (-r_1 r_2, r_1 \odot e_2, r_2 \odot e_1) = (-r_1 r_2, e_2^{r_1}, e_1^{r_2}).$$

We have $\boldsymbol{\lambda} \odot \Gamma = \Theta(C)$ if and only if $e(h_1, h_2)^{-r_1 r_2} \cdot e(h_1, e_2^{r_1}) \cdot e(e_1^{r_2}, h_2) = e(e_1, e_2)/B$, and thus,

$$\begin{aligned} B &= e(e_1, e_2)/(e(h_1^{r_1}, X_2) \cdot e(e_1, h_2^{r_2})) \\ &= e(e_1, X_2)/e(h_1^{r_1}, X_2) = e(X_1, X_2) \end{aligned}$$

For the sake of completeness, if $\gamma_1 = e(g_1, g_2)^{\alpha_1} e(h_1, h_2)^{\alpha_4}$, $\gamma_2 = g_1^{\alpha_2} h_1^{\alpha_4}$, and $\gamma_3 = g_2^{\alpha_3} h_2^{\alpha_4}$, the corresponding hash value is:

$$H = e(u_1, u_2)^{-\alpha_1} \cdot e(u_1, e_2)^{\alpha_2} \cdot e(e_1, u_2)^{\alpha_3} \cdot (e(e_1, e_2)/B)^{\alpha_4} = \gamma_1^{-r_1 r_2} \cdot e(\gamma_2, e_2^{r_1}) \cdot e(e_1^{r_2}, \gamma_3).$$

Notations. Let us now introduce notation to handle any quadratic equation. In addition to previous notations, as in Section D.4, we also write $\mathsf{ek}_T = (g_T, h_T = g_T^{x_T})$ a public key for ElGamal encryption scheme in $\mathbb{G}_T$. We are interested in languages of commitments $(C_{1,i})_i$ of $(X_{1,i})_i \in \mathbb{G}_1^{n_1}$, $(C_{2,j})_j$ of $(X_{2,j})_j \in \mathbb{G}_2^{n_2}$, and $(C_{T,k})_i$ of $(X_{T,k})_k \in \mathbb{G}_T^{n_T}$ such that:

$$\prod_i e(X_{1,i}, A_{2,i}) \cdot \prod_j e(A_{1,j}, X_{2,j}) \cdot \prod_i \prod_j e(X_{1,i}, X_{2,j})^{a_{i,j}} \cdot \prod_k X_{T,k}^{a_{T,k}} = B, \tag{4}$$

with $\mathsf{aux} = ((A_{2,i})_i, (A_{1,j})_j, (a_{i,j})_{i,j}, (a_{T,k})_k) \in \mathbb{G}_1^{n_1} \times \mathbb{G}_2^{n_2} \times \mathbb{Z}_p^{n_1 n_2 + n_T}$. This can also be written:

$$\left(\bigoplus_{i=1}^{n_1} A_{2,i} \odot X_{1,i}\right) \oplus \left(\bigoplus_{j=1}^{n_2} A_{1,j} \odot X_{2,j}\right) \oplus \left(\bigoplus_{i=1}^{n_1} \bigoplus_{j=1}^{n_2} a_{i,j} \odot X_{1,i} \odot X_{2,j}\right) \oplus \left(\bigoplus_{k=1}^{n_T} a_{T,k} \odot X_{T,k}\right) = B.$$

Let us also write, for any $\omega \in \{1, 2, T\}$ and $\iota \in \{1, \ldots, n_\omega\}$: $C_{\omega,\iota} = (u_{\omega,\iota} = g_\omega^{r_{\omega,\iota}}, e_{\omega,\iota} = h_\omega^{r_{\omega,\iota}} X_{\omega,\iota})$.

Basic Scheme in $\mathbb{G}_T$. Let us consider the following matrix, with a diagonal block

$$\Gamma = \begin{pmatrix} g_1 \odot g_2 & 1 & 1 & 1 & h_1 \odot h_2 \\ 1 & g_1 & 1 & 1 & h_1 \\ 1 & 1 & g_2 & 1 & h_2 \\ 1 & 1 & 1 & g_T & h_T \end{pmatrix}$$

With

$$\Theta(C) = \begin{pmatrix} \bigoplus_i \bigoplus_j -a_{i,j} \odot u_{1,i} \odot u_{2,j}, \left(\bigoplus_i \bigoplus_j a_{i,j} \odot u_{1,i} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot u_{1,i}\right), \\ \left(\bigoplus_i \bigoplus_j a_{i,j} \odot e_{1,i} \odot u_{2,j}\right) \oplus \left(\bigoplus_j A_{1,j} \odot u_{2,j}\right), \bigoplus_i a_{T,i} \odot u_{T,i}, \\ \left(\bigoplus_i \bigoplus_j a_{i,j} \odot e_{2,i} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot e_{1,i}\right) \oplus \left(\bigoplus_j A_{1,j} \odot e_{2,j}\right) \oplus \left(\bigoplus_k a_{T,k} \odot e_{T,k}\right) \ominus B \end{pmatrix}$$

the requirement $\boldsymbol{\lambda} \odot \Gamma = \Theta(C)$ implies

$$\begin{aligned} \boldsymbol{\lambda} &= \begin{pmatrix} \bigoplus_i \bigoplus_j -a_{i,j} \odot r_{1,i} \odot r_{2,j}, \left(\bigoplus_i \bigoplus_j r_{1,i} \odot a_{i,j} \odot e_{2,j}\right) \oplus \left(\bigoplus_i A_{2,i} \odot r_{1,i}\right), \\ \left(\bigoplus_i \bigoplus_j r_{2,i} \odot a_{i,j} \odot e_{1,j}\right) \oplus \left(\bigoplus_j A_{1,j} \odot r_{2,j}\right), \bigoplus_k r_{T,k} \end{pmatrix} \\ &= \left(\textstyle\sum_i \sum_j a_{i,j} r_{1,i} r_{2,j}, \prod_i \prod_j e_{2,j}^{r_{1,i} a_{i,j}} \cdot \prod_i A_{2,i}^{r_{1,i}}, \prod_i \prod_j e_{1,j}^{r_{2,i} a_{i,j}} \cdot \prod_j A_{1,j}^{r_{2,j}}, \textstyle\sum_k r_{T,k}\right), \end{aligned}$$

and it is satisfied, if and only if Equation (4) is true, *i.e.*, the word is in the language.

Variant. The same trick as the one used in the variant of the SPHF for linear pairing equation can be used to avoid having too many elements of the projection key in $\mathbb{G}_T$.