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

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

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

Abstract. *Password-Authenticated Key Exchange* (PAKE) has received deep attention in the last few years, with a recent strong improvement by Katz-Vaikuntanathan, and their one-round protocol: 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. We follow their work with a further study of their new *Smooth-Projective Hash Function* framework, and namely we introduce new efficient instantiations on IND-CCA ciphertexts. It allows us to design 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. Our scheme resists off-line dictionary attacks in the Bellare-Pointcheval-Rogaway model, under the DDH assumption with a CRS.

We thereafter show how our new instantiations can prove more complex equations. We then apply them to propose quite efficient instantiations in the standard model of the more general family of protocols, termed *Langage-Authenticated Key Exchange* (LAKE) by Blazy-Chevalier-Pointcheval-Vergnaud. They include quite concrete key exchange protocols, such as PAKE, Verifier-based PAKE and Secret Handshakes. In Verifier-based PAKE, the server knows a transformation of the password only, which limits impact of the corruption of the server, since exhaustive search would still have to be performed to recover the actual passwords. In Secret Handshakes, two members of the same group want to identify each other secretly, in the sense that each party reveals his affiliation to the other only if they are members of the same group. Outsiders do not learn anything about the outcome of the protocol.

Keywords. Authenticated Key Exchange, Commitments, Smooth Projective Hash Functions

1 Introduction

Authenticated Key Exchange protocols are quite important primitives in practice, since they enable two parties to generate a shared high entropy secret key, to be later used with symmetric primitives to protect communications, while interacting over an insecure network, under the control of the adversary.

Various means have been proposed to allow the two parties to authenticate to each other, and the most practical one is definitely a shared low entropy secret, or a so-called password they can agree on over the phone. The most famous instantiation has been proposed by Bellovin and Merritt [BM92], EKE for Encrypted Key Exchange, which simply consists in a Diffie-Hellman key exchange [DH76], where the flows are symmetrically encrypted under the shared password. After the proposal of a formal security model (BPR) [BPR00], several proofs and variants of EKE have been proposed in the ideal-cipher model or the random-oracle model. Katz, Ostrovsky and Yung [KOY01] proposed the first practical scheme (KOY), provably secure in the standard model. It has been generalized by Gennaro and Lindell [GL03], making use of smooth projective hash functions.

Smooth Projective Hash Functions (SPHF) have been introduced by Cramer and Shoup [CS02] in order to build IND-CCA encryption schemes [CS98]. They can be seen as a kind of designated-verifier proofs of membership. The notion of SPHF has thereafter been extended to a more general context, by Gennaro and Lindell [GL03], in order to build *Password-Authenticated Key Exchange* (PAKE) protocols, secure against dictionary attacks, in the same vein as the KOY protocol [KOY01], in the CRS framework. This approach has thereafter been applied to the *Universal Composability* (UC) framework by Canetti, Halevi, Katz, Lindell and MacKenzie [CHK$^+$05], and improved by Abdalla, Chevalier and Pointcheval [ACP09] to resist adaptive corruptions. SPHF have thus been shown to have quite rich applications, and namely to conditioned or authenticated actions [BPV12, BCPV12].

† CNRS – UMR 8548 and INRIA – EPI Cascade

More recently, the ultimate step for PAKE has been achieved by Katz-Vaikuntanathan [KV11], who propose a practical one-round PAKE, where the two players just have to send simultaneous flows to each other, that depend on their own password only. Because of the simultaneous flows, one flow cannot depend on the other one, which makes impossible the use of standard SPHF. This led Katz-Vaikuntanathan to provide a new definition for 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 (more details follow). The construction in the BPR model is quite efficient, since it just consists in sending an IND-CCA ciphertext of the password and a SPHF projection key, for the correctness of the partner's ciphertext, in each direction. The shared secret key is eventually the product of the two hash values. The instantiation proven under the DLin assumption uses a ciphertext consisting of 66 group elements and a projection key consisting of 4 group elements. As a consequence, after having sent 70 group elements each, the two players agree on a shared high-entropy secret. In [LY12], Libert and Yung proposed a new technique for threshold encryption scheme that can be applied to have a more efficient simulation-sound proof of plaintext equality in the Naor-Yung-type cryptosystem in [KV11]: 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 users. Jutla-Roy [JR12] proposed a new publicly-verifiable IND-CCA encryption to achieve a better efficiency in their protocol.

1.1 Our Results

We revisit the different definitions for SPHF proposed in [CS02, GL03, KV11], denoted respectively CS−SPHF, GL−SPHF and KV−SPHF. We propose several new instantiations, which are significatively more efficient. Our first instantiation on a Cramer-Shoup ciphertext leads to the most efficient PAKE in the BPR security model, based on the sole DDH assumption: it relies on the above Katz-Vaikuntanathan [KV11] framework, but without any simulation-sound NIZK to build an IND-CCA ciphertext, since we can directly use the Cramer-Shoup encryption scheme [CS98] and not the Naor-Yung paradigm [NY90]. Our scheme just consists of 6 group elements in each direction under the DDH assumption (4 elements for the Cramer-Shoup ciphertext, and 2 for the projection key). A DLin variant also exists and consists of 9 group elements in each direction (5 elements for the Linear Cramer-Shoup ciphertext, and 4 for the projection key) also without any pairing requirement, to be compared with the 70 group elements for Katz-Vaikuntanathan [KV11] and the 20 group elements for Jutla-Roy's improvement [JR12] which both need pairings.

We extend this technique to handle Multi-Exponentiations Equations with variables in $\mathbb{G}$ without any pairing requirements to be at least as general as the Groth-Sahai methodology [GS08]. We thereafter show (in the Appendix) how SPHF can be used to handle more general pairing product equations than those from Groth-Sahai. We can then apply them to the *Language-Authenticated Key-Exchange* (LAKE) protocols introduced in [BCPV12], which are extensions of *Credential-Authenticated Key-Exchange* (CAKE) [CCGS10].

1.2 Our Techniques

Our methodology uses SPHF, as initially proposed by Cramer-Shoup (CS−SPHF) [CS02], thereafter extended by Gennaro-Lindell (GL−SPHF) [GL03] and more recently by Katz-Vaikuntanathan (KV−SPHF) [KV11]. Basically, SPHF are families of pairs of functions $(\mathsf{Hash}, \mathsf{ProjHash})$ defined on a language L. These functions are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where hk, the hashing key, can be seen as a 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 $W \notin \mathrm{L}$, this witness does not exist, and the smoothness property states that $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$ is independent of hp, and thus even knowing hp, one cannot guess $\mathsf{Hash}(\mathsf{hk}, \mathrm{L}, W)$. Variations between CS−SPHF, GL−SPHF and KV−SPHF rely in the way one computes hp from hk and possibly W, but also on the smoothness property, according to the freedom the adversary has to choose W (before or after having seen hp).

The strongest definition is the recent KV−SPHF [KV11], where hp depends on hk only, and the smoothness should hold even if the adversary can choose W after having seen hp. Thereafter, Katz-Vaikuntanathan [KV11] wanted to

build such KV−SPHF on the language of the valid ciphertexts of a given message m, under an IND-CCA encryption scheme. Previous SPHF known for Cramer-Shoup ciphertexts were GL−SPHF only, because the projection key hp had to be computed for a specific word W only, and thus depended on both hk and W. Katz-Vaikuntanathan thus had to use another IND-CCA encryption scheme. For this reason, they applied the Naor-Yung methodology [NY90] on the ElGamal [ElG85] or the Linear encryption [BBS04], with a simulation-sound non-interactive zero-knowledge proof [GS08].

In this paper, we show how to build KV−SPHF on Cramer-Shoup [CS98] and Linear Cramer-Shoup [Sha07] ciphertexts, which lead to much more efficient schemes.

1.3 Language Definition

We then consider more complex languages than simple ciphertexts of a known message: we extend our SPHF to ciphertexts of messages that satisfy a specific relation. We use the more general formalism defined in [BCPV12]: for any efficiently computable binary relation $\mathcal{R} : \{0,1\}^* \times \mathcal{P} \times \mathcal{S} \to \{0,1\}$, where the parameters $\mathsf{pub} \in \{0,1\}^*$ and $\mathsf{priv} \in \mathcal{P}$ define the language $\mathrm{L}(\mathsf{pub}, \mathsf{priv}) \subseteq \mathcal{S}$ of the words W such that $\mathcal{R}((\mathsf{pub}, \mathsf{priv}), W) = 1$, and thus, which can be efficiently checked. More precisely, pub are public parameters, priv are private parameters everybody has in mind, and W is a word in the language. The values pub and priv can be seen as auxiliaries inputs: $\mathsf{aux} = (\mathsf{pub}, \mathsf{priv})$. We will then use the notation $\mathrm{L}_{\mathsf{aux}}$ for the language of the words W that satisfy a fixed (well-defined) relation with parameters aux. In the following, the above word W will be committed to, but never revealed, so we can define companion relation and language: $\mathcal{CR}(\mathsf{aux}, C, w)$, where aux contains the auxiliary (public and private) parameters, C is a commitment of a word W under the random coins r and $w = (r, W)$, such that $\mathcal{CR}(\mathsf{aux}, C, w) = 1 \iff \mathcal{R}(\mathsf{aux}, W) = 1$; and $\textsc{Lofc}_{\mathsf{aux}}$, the language of the ciphertexts C of plaintext W under random coins r that satisfy $\mathcal{CR}(\mathsf{aux}, C, (r, W)) = 1$. The words in the new language $\textsc{Lofc}_{\mathsf{aux}}$ are actually ciphertexts, and the random coins and the plaintext are the witnesses. Our formalism covers languages on ciphertexts of messages that satisfy $\mathcal{P}$ languages only, but it is enough for our purpose.

In [BCPV12], the authors defined the LAKE primitive, specific to two relations. It allows two users, Alice and Bob, each owning a word in a specific language, to agree on a shared high entropy secret if the other actually knows a word in the language that one thinks about: they first both agree on the public parameter pub, but Bob will think about $priv_a$ for his expected Alice's value of priv_a, Alice will do the same with $priv_b$ for priv_b; eventually, if $priv_a = \mathsf{priv}_a$ and $priv_b = \mathsf{priv}_b$, and if they both know words in the languages, then the key agreement will succeed. In case of failure, no information should leak about the reason of failure, except the inputs did not satisfy the relations, or the languages were not consistent. Thanks to our new KV−SPHF on ciphertexts, we will build efficient LAKE protocols with provable security in the standard model.

1.4 Outline of the Paper

In Section 2, we give an informal definition of SPHF, with several instantiations on Cramer-Shoup ciphertexts. We then show our main application: a highly-efficient one-round PAKE. In Section 3, we state more formal definitions of SPHF, and constructions for more general languages are provided in Section 4. We then apply that to Language-Authenticated Key Exchange in the standard model, in Section 5. More details can also be found in the Appendix.

2 A new SPHF on Cramer-Shoup Ciphertexts and its Application

In this section, after recalling the Cramer-Shoup commitment scheme, we introduce a new SPHF for Cramer-Shoup ciphertexts/commitments. We then show one of its applications: a very efficient instantiation of the one-round PAKE of Katz-Vaikuntanathan [KV11].

2.1 Cramer-Shoup Commitment Scheme

Commitment Scheme. Commitments allow a user to commit to a value, without revealing it, but without the possibility to later change his mind. It is composed of three algorithms, defined as follows for the labeled-version: $\mathsf{Setup}(1^{\mathfrak{K}})$ generates the system parameters, according to a security parameter $\mathfrak{K}$; $\mathsf{Commit}(\ell, m; r)$ produces a commitment c and the opening information d on the input message $m \in \mathcal{M}$ using the random coins r, under the label ℓ; while $\mathsf{Decommit}(\ell, c, m, d)$ checks whether c is a valid commitment of m according to opening information d.

Such a commitment scheme should be both *hiding*, which says that the commit phase does not leak any information about m, and *binding*, which says that the decommit phase should not be able to open to two different messages. Additional features will be required in the sequel, such as non-malleability and extractability. We also included a label ℓ, which can be empty or an additional public information that has to be the same in both the commit and the decommit phases. A labeled commitment that is both non-malleable and extractable can be instantiated by an IND-CCA labeled encryption scheme [GL03].

Cramer-Shoup Commitment Scheme (CSCom). The CS encryption scheme is recalled in the Appendix A.4. We thus briefly review the commitment version. The parameters, in the CRS, are a group $\mathbb{G}$ of prime order p, with two independent generators $(g_1, g_2) \stackrel{\$}{\leftarrow} \mathbb{G}^2$, a hash function $\mathfrak{H}_K \stackrel{\$}{\leftarrow} \mathcal{H}$ 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 $(x_1, x_2, y_1, y_2, z) \stackrel{\$}{\leftarrow} {\mathbb{Z}_p}^5$, 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 public parameters consist of the encryption key $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h, \mathfrak{H}_K)$, while the trapdoor for extraction is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z)$. One can define the commitment process on the message $m \in \{0,1\}^n$ as the Cramer-Shoup encryption, where $M = \mathcal{G}(m) \in \mathbb{G}$ and $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e) \in \mathbb{Z}_p^*$:

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

In case one wants to encrypt a vector of messages, encoded as group elements $(M_1, \ldots, M_n)$, all at once in a non-malleable way, one computes, with a common $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$,

$$C \stackrel{\text{def}}{=} \mathsf{CS}(\ell, \mathsf{ek}, M_1, \ldots, M_n; r) \stackrel{\text{def}}{=} (\ell, \{\mathbf{u}_i = (g_1^{r_i}, g_2^{r_i}), e_i = M_i \cdot h^{r_i}, v_i = (cd^{\xi})^{r_i}\}_{i=1,\ldots,n}).$$

This can be seen as the vector of the individual message ciphertexts, with the same label ℓ and the same ξ value. Hence, everything done on tuples of ciphertexts will work on ciphertexts of vectors.

2.2 Informal Definitions of SPHF

Let us consider a family of language $\mathrm{L}_{\mathsf{aux}}$ indexed by aux, as described in Section 1.3. In the main part of this article, elements $\mathrm{L}_{\mathsf{aux}}$ are always tuples of group elements. We are interested in languages of tuples of ciphertexts $C = (C_1, \ldots, C_n)$ whose corresponding plaintexts $W = X = (X_1, \ldots, X_n)$ are in $\mathrm{L}_{\mathsf{aux}}$. More formally, let us consider the following language family, where $\ell_1, \ldots, \ell_n$ are some public labels:

$$\textsc{Lof}\mathrm{C}_{\mathsf{aux}} = \left\{ C = (C_1, \ldots, C_n) \;\middle|\; \begin{array}{l} \exists r_1, \ldots, r_n, X_1, \ldots, X_n, \\ C_1 = \mathsf{Commit}(\ell_1, X_1; r_1), \ldots, C_n = \mathsf{Commit}(\ell_n, X_n; r_n) \\ \text{and } (X_1, \ldots, X_n) \in \mathrm{L}_{\mathsf{aux}} \end{array} \right\}.$$

A smooth projective hash function (SPHF) system for the language $\textsc{Lof}\mathrm{C}_{\mathsf{aux}}$, whatever the exact model, is defined with five algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates the system parameters, according to a security parameter $\mathfrak{K}$;
- $\mathsf{HashKG}(\mathsf{aux})$ generates a hashing key hk for the language $\textsc{Lof}\mathrm{C}_{\mathsf{aux}}$;
- $\mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, C)$ derives the projection key hp, possibly depending on the word C;
- $\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, C)$ outputs the hash value from the hashing key;
- $\mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, C, w)$ outputs the hash value from the projection key hp, and the witness w, which consists of the random coins and the plaintexts: $w = (r_1, \ldots, r_n, X_1, \ldots, X_n)$, that $C \in \textsc{Lof}\mathrm{C}_{\mathsf{aux}}$.

The correctness of SPHF assures that if C is in $\text{LOF}\text{C}_{\mathsf{aux}}$ with w a witness of this membership, and thus the plaintext $X = (X_1, \ldots, X_n)$ is in the companion language L_{aux}, then the two ways to compute the hash values give the same result: $\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, C) = \mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, C, w)$. In our setting, these hash values will belong to a group $\mathbb{G}$, denoted multiplicatively. Informally, the security is defined through the smoothness, which guarantees that if $C \notin \text{LOF}\text{C}_{\mathsf{aux}}$, the hash value is *statistically* indistinguishable from a random element, even knowing hp.

In this section, we are interested in the very strong notion of SPHF, defined by Katz-Vaikuntanathan in [KV11]: while the GL−SPHF definition allowed the projection key hp to depend on the word C, the KV−SPHF definition limits, as in the original CS−SPHF definition, the projection key hp not to depend on C, and requires, in addition, the smoothness to 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 he is interested in.

We now exhibit our new SPHF and show its application to the one-round PAKE.

2.3 The GL−SPHF for CSCom

Gennaro-Lindell [GL03] proposed an SPHF on a labeled Cramer-Shoup ciphertext: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^4$. The associated projection key, on a ciphertext $C = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r)$, is $\mathsf{hp} = g_1^{\eta_1} g_2^{\eta_2} h^\lambda (cd^\xi)^\mu \in \mathbb{G}$. Then, one can compute the hash value in two different ways, for the language LOFC_m of the valid ciphertexts of m, with $M = \mathcal{G}(m)$:

$$H \stackrel{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, m, C) \stackrel{\text{def}}{=} u_1^{\eta_1} u_2^{\eta_2} (e/M)^\lambda v^\mu = \mathsf{hp}^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, m, C, r) \stackrel{\text{def}}{=} H'.$$

Since the projection key depends on C, or at least on ξ which can be computed when C is known only, Katz-Vaikuntanathan [KV11] had to use another IND-CCA encryption scheme to have a KV−SPHF for their PAKE application.

2.4 A KV−SPHF for CSCom

Here is the description of our new KV−SPHF for labeled Cramer-Shoup ciphertexts: the hashing key just consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$. The associated projection key is

$$\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\lambda c^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d^\mu) \in \mathbb{G}^2.$$

Then one can compute the hash value in two different ways, for the language LOFC_m of the valid ciphertexts of m, with $M = \mathcal{G}(m)$:

$$H \stackrel{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, m, C) \stackrel{\text{def}}{=} u_1^{(\eta_1 + \xi\eta_2)} u_2^\theta (e/M)^\lambda v^\mu = (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, m, C, r) \stackrel{\text{def}}{=} H'.$$

Let us claim and prove the security properties (formal definitions are given in the next section):

Theorem 1. *The above* SPHF *is a* 0*-smooth (or perfectly smooth)* KV−SPHF.

Proof. Let us suppose C is not a correct encryption of m. Since $\xi \in \mathbb{Z}_p^*$, there exists $(s, t) \in \mathbb{Z}_p^2$ and $M' \in \mathbb{G}$ such that:

$$C = (\ell, \mathbf{u} = (g_1^r, g_2^s), e = h^r \cdot M', v = c^r d^{\xi t})$$

with either $M \neq M'$, $s \neq r$ or $t \neq r$. We want to prove that, in this case, hp gives no information on the value of H. Let us write $g_2 = g_1^\beta$, $c = g_1^\gamma$, $d = g_1^\delta$ and $\Delta = \log_{g_1}(M'/M)$. We have $\beta \neq 0$, $\gamma \neq 0$ and $\delta \neq 0$. We can remark that:

$$\begin{pmatrix} \log_{g_1} \mathsf{hp}_1 \\ \log_{g_1} \mathsf{hp}_2 \\ \log_{g_1} H \end{pmatrix} = \begin{pmatrix} 1 & 0 & \beta & z & \gamma \\ 0 & 1 & 0 & 0 & \delta \\ r & \xi r & s\beta & rz + \Delta & r\gamma + \xi t\delta \end{pmatrix} \cdot \begin{pmatrix} \eta_1 & \eta_2 & \theta & \lambda & \mu \end{pmatrix}^T$$

and that H is completely unpredictable given hp (*i.e.*, its distribution is perfectly uniform given hp), if the last line L_3 of the matrix is linearly independent from the two first ones L_1 and L_2. Therefore, because of the two first columns, H is unpredictable unless the line vector $L_3 - rL_1 - r\xi L_2 = (0, 0, (s - r)\beta, \Delta, \xi(t - r)\delta)$ is zero. As a consequence, unless $s = r$, $\Delta = 0$ and $t = r$, (*i.e.*, C is a valid ciphertext of m), H is unpredictable. □

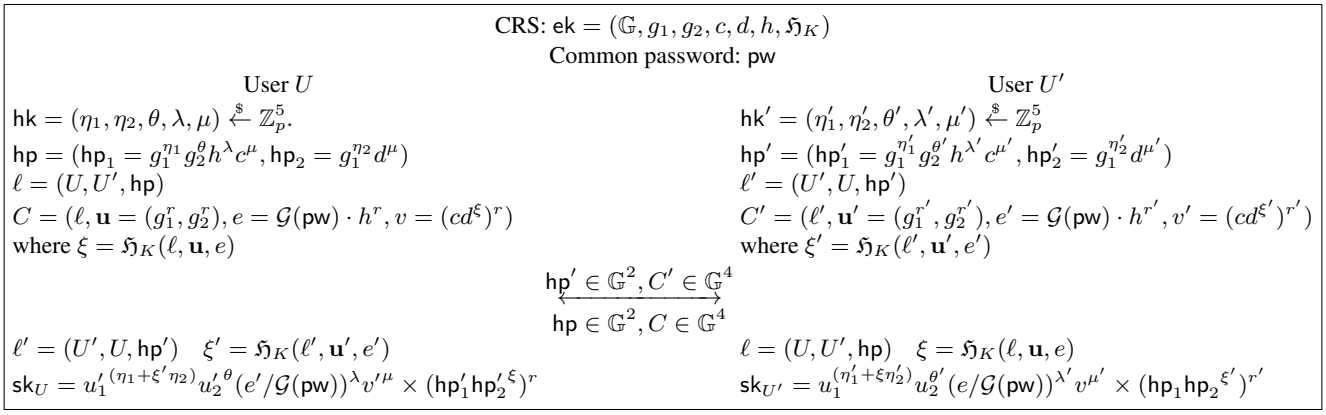

Fig. 1. One-Round PAKE based on DDH

2.5 Application: Efficient One-Round PAKE

We can thus use a Cramer-Shoup commitment and our new KV−SPHF to instantiate the framework for one-round PAKE of Katz-Vaikuntanathan [KV11], instead of a double ElGamal or Linear encryption plus a simulation-sound non-interactive zero-knowledge proof.

This yields a very efficient one-round PAKE: in our DDH instantiation, each user only sends 6 elements of $\mathbb{G}$ (see Figure 1), instead of 70 elements of $\mathbb{G}$ for their DLin instantiation, which they claim to be the most efficient thanks to a specific simulation-sound non-interactive zero-knowledge proof, using the Groth-Sahai [GS08] methodology. Using a relatively-sound non-interactive zero-knowledge proof, Jutla-Roy [JR12] proposed a new encryption scheme to instantiate the framework and reduced the communication complexity to 20 group elements for each user.

We can also easily adapt our construction to be secure under the DLin assumption. In this case, each user sends 9 elements of $\mathbb{G}$ (the ciphertext C is composed of 5 elements and hp is composed of 4 elements), to be compared against their instantiation that requires 70 group elements in each direction. Details can be found in the Appendix B.1. We insist that – contrary to [KV11, JR12] – in both DDH and DLin versions no pairing computations are required. Any group where the DDH or the DLin problems are hard can be used without the need of any pairing-friendly environments.

For the sake of completeness, we also present a DLin version in the Appendix B.1 of this one-round PAKE in a pairing environment allowing to reduce the number of group element to 7 for each users. We also propose a fairly efficient instantiation of the UC version of this PAKE in Appendix D, following the Katz-Vaikuntanathan [KV11] framework, but using our above improvement to the non-UC scheme, and a new efficient Simulation-Sound Extractable Non-Interactive Zero-Knowledge proof. The resulting scheme is significantly more efficient than the previous proposals [KV11, JR12].

3 Formal Definitions

In this section, we propose two new orthogonal classifications for SPHF. First, we present our classification of SPHF based on the dependence between words and keys. According to this classification, there are three types of SPHF: Gennaro-Lindell [GL03] type (GL−SPHF), Cramer-Shoup [CS02] type (CS−SPHF), and Katz-Vaikuntanathan [KV11] type (KV−SPHF). We then present our second classification of SPHF based on the dependence between languages (auxiliary inputs) and keys.

We use the classical notions of statistical and computational distances recalled in the Appendix A.3 and we say that two distributions are ε-close if their statistical distance is at most ε. In the following, we will focus on languages of ciphertexts, where the witnesses are the random coins used for encryption. Hence, we will use the notation of $\text{LOFC}_{\mathsf{aux}}$ for the languages and the words will be denoted C. We assume that these languages $\text{LOFC}_{\mathsf{aux}}$ are included in a set $\mathcal{S}et$

and we denote $\overline{\text{LOFC}}_{\mathsf{aux}} = \mathcal{S}et \setminus \text{LOFC}_{\mathsf{aux}}$. Finally, we denote by Π the set of hash values (in most of the concrete instantiations, this will be a multiplicative group).

3.1 Smoothness Adaptivity and Key Word-Dependence

KV−SPHF. A KV−SPHF is the strongest SPHF: 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 HashKG does not use its input C and if, for any aux and for any function f onto $\overline{\text{LOFC}}_{\mathsf{aux}}$, the two following distributions are ε-close:

$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{aux});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, \bot);\ C \leftarrow f(\mathsf{hp});\ H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, C)\}$$
$$\{(\mathsf{hp}, H) \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{aux});\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, \bot);\ C \leftarrow f(\mathsf{hp});\ H \xleftarrow{\$} \Pi\}.$$

CS−SPHF. A CS−SPHF is weaker than a KV−SPHF: the projection key hp still does not depend on the word C (word-independent key), but it is no more required for the smoothness to hold even if C depends on hp (non-adaptive smoothness). More formally, a CS−SPHF is ε-smooth, if for any aux and for any $C \in \overline{\text{LOFC}}_{\mathsf{aux}}$, the two following distributions are ε-close:

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

It is clear to 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 choosing C adaptively does not increase the distance. However, as soon as a weak word C can bias the distribution, f can exploit it, and, as will be shown in Section 4.3, there exist CS−SPHF which are not KV−SPHF.

GL−SPHF. A GL−SPHF is the weakest SPHF: the projection key hp depends on the word C (word-dependent key). More formally, a GL−SPHF is ε-smooth, if for any aux and for any $C \in \overline{\text{LOFC}}_{\mathsf{aux}}$, the two following distributions are ε-close:

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

3.2 Key Language-Dependence

In this section, we additionally classify SPHF based on the dependence between aux and keys hk and hp. There are three types: type I, type II, type III.

Type I (Language-Dependent Key). A type I SPHF is the weakest type of SPHF. It just corresponds to the original definition of Section 2.2, without any extra property, where the keys are computed from aux.

Type III (Language-Independent Key). A type III SPHF is the strongest type of SPHF in this classification: algorithms HashKG and ProjKG do not use their input aux. This means, in particular, that a projection key hp can be computed before knowing aux, which can be very useful for LAKE constructions, with precomputations (see Section 5).

Type II (Language-Indistinguishable Key). A type II SPHF is slightly weaker than a type III SPHF: algorithms HashKG and ProjKG may use their input aux but hp looks independent from aux. More formally, an SPHF is of type II if, for any $\mathsf{aux}_1, \mathsf{aux}_2$, and any word C, the two following distributions are computationally/statistically/perfectly indistinguishable:

$$\{\mathsf{hp} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{aux}_1);\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}_1, C)\}$$
$$\{\mathsf{hp} \mid \mathsf{hk} \xleftarrow{\$} \mathsf{HashKG}(\mathsf{aux}_2);\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}_2, C)\}.$$

4 Constructions

In this section, after a brief introduction and after recalling previous constructions of GL−SPHF, we first present various novel constructions of SPHF, for languages defined by ciphertexts (of CS encryption scheme) which plaintext satisfy a system of multi-exponentiation equations: a KV−SPHF, which is an extension of the construction of Section 2, a very efficient CS−SPHF, using only 3 group elements and one number roughly of size $\mathfrak{K}$ (the security parameter) and an even more efficient GL−SPHF, using only 2 group elements and one $\mathfrak{K}$-bit long integer. Then, we show how reusing randomness in CSCom commitments can reduce the total amount of group elements. Eventually, we briefly show that these results also apply to ElGamal, Linear, and Linear Cramer-Shoup commitments, and we present a summary of our various SPHF. Due to space limitation, all proofs can be found in the Appendix B.2.

In the Appendix B.3, we also show how to use these SPHF for languages defined by ciphertexts which plaintext satisfy a system of quadratic pairing product equations. To this aim, we will need additional ciphertexts, and we will thus modify a little bit the languages.

4.1 Introduction

System of Multi-Exponentiation Equations over CSCom. In all the Section 4, our constructions are in a cyclic group $\mathbb{G}$ of prime order p. Constructions in pairing-friendly environments will be considered in the Appendix B.3 only. All our constructions enable us to deal with any system of multi-exponentiation equations over committed values using a CSCom. Formally, let $C_1, \ldots, C_n$ be commitments of values $X_1, \ldots, X_n$ in $\mathbb{G}$, with label $\ell_1, \ldots, \ell_n$. They can be written as $C_i = (\ell_i, \mathbf{u}_i = (g_1^{r_i}, g_2^{r_i}), e_i = X_i \cdot h^{r_i}, v_i = (cd^{\xi_i})^{r_i})$, with $\xi_i = \mathfrak{H}_K(\ell_i, \mathbf{u}_i, e_i)$. A system of multi-exponentiation equations is a system of t equations of the form:

$$\prod_{i \in I_1} X_i^{a_{1,i}} = A_1 \qquad (1.1) \qquad \cdots \qquad \prod_{i \in I_t} X_i^{a_{t,i}} = A_t \qquad (1.t)$$

with $I_k \subseteq \{1, \ldots, n\}$, $a_{k,i} \in \mathbb{Z}_p$ and $A_i \in \mathbb{G}$. We suppose that $a_{k,i} = 0$ for $i \notin I_k$, and $a_{k,i} \neq 0$ for $i \in I_k$. We can suppose that the CSCom parameters, as well as t and n are fixed, but $\mathsf{aux} = (a_{k,i})_{k \in \{1,\ldots,t\}, i \in \{1,\ldots,n\}} \cup (A_k)_{k \in \{1,\ldots,t\}}$ will fully define the language $\textsc{Lofc}_{\mathsf{aux}}$. Our three first SPHF constructions enable to prove that commitments $C_1, \ldots, C_n$ are valid commitments (*i.e.*, are not rejected by the CS decryption algorithm) and that $X_1, \ldots, X_n$ verify equations (1.1), ..., (1.t). Recall that if one wants global non-malleability, one can use a global ξ value, and even a unique label, as explained in Section 2.1.

Previous Constructions. Constructions of SPHF for linear pairing equations can be found in [BCPV12]. However these constructions are only (type III) GL−SPHF constructions and are much less efficient than our new GL−SPHF constructions.

4.2 KV−SPHF Constructions

We first propose a type III scheme, and then a much more efficient type II scheme.

Type III Scheme. The hashing key hk is a list of n scalar 4-tuples $\mathsf{hk}_{1,i} = (\eta_{1,i}, \eta_{2,i}, \theta_i, \mu_i)$, for $i \in \{1, \ldots, n\}$, and of t scalar couples $\mathsf{hk}_{3,k} = (\kappa_k, \lambda_k)$, for $k \in \{1, \ldots, t\}$. The projection key hp is composed of $2n$ projection sub-keys $\mathsf{hp}_{1,i}$, $\mathsf{hp}_{2,i}$, and t projection sub-keys $\mathsf{hp}_{3,k}$ with:

$$\mathsf{hp}_{1,i} = g_1^{\eta_{1,i}} g_2^{\theta_i} c^{\mu_i} \qquad \mathsf{hp}_{2,i} = g_1^{\eta_{2,i}} d^{\mu_i} \qquad \mathsf{hp}_{3,k} = g_1^{\kappa_k} h^{\lambda_k}.$$

The $\mathsf{hk}_{1,i}$, $\mathsf{hp}_{1,i}$ and $\mathsf{hp}_{2,i}$ parts are used to ensure commitments are valid and the $\mathsf{hk}_{3,k}$ and $\mathsf{hp}_{3,k}$ parts are used to ensure equations are verified. The hash value is:

$$H \stackrel{\mathsf{def}}{=} \prod_{i=1}^{n} \left(u_{1,i}^{\eta_{1,i}+\xi_i\eta_{2,i}} u_{2,i}^{\theta_i} v_i^{\mu_i}\right) \times \prod_{k=1}^{t} \left(\prod_{i\in I_k} \left(u_{1,i}^{a_{k,i}\kappa_k} e^{a_{k,i}\lambda_k}\right) A_k^{-\lambda_k}\right) = \prod_{i=1}^{n} \left(\mathsf{hp}_{1,i}\mathsf{hp}_{2,i}^{\xi_i}\right)^{r_i} \times \prod_{k=1}^{t} \prod_{i\in I_k} \mathsf{hp}_{3,k}^{a_{k,i}r_i} \stackrel{\mathsf{def}}{=} H'.$$

Theorem 2. *The above* SPHF *is a type III* 0*-smooth (or perfectly smooth)* KV$-$SPHF.

Type II Scheme. In a type II scheme, hp can depend on $a_{k,i}$'s, which makes hp specific to the system of multi-exponentiation equations but still independent of the commitments. This relaxation enables us to reduce the size of hp and hk, by mixing all the $\mathsf{hp}_{3,k}$ with the $\mathsf{hp}_{1,i}$: $\mathsf{hk}_{3,k}$ are all reduced to (λ_k) (κ_k are removed), for $k \in \{1, \ldots, t\}$. The projection key hp does not contain $\mathsf{hp}_{3,k}$ any more and $\mathsf{hp}_{1,i}$ is changed to $\mathsf{hp}_{1,i} = g_1^{\eta_{1,i}} g_2^{\theta_i} h^{\sum_{k=1}^{t} a_{k,i}\lambda_k} c^{\mu_i}$, for $i \in \{1, \ldots, n\}$. The hash value is:

$$H \stackrel{\mathsf{def}}{=} \prod_{i=1}^{n} \left(u_{1,i}^{\eta_{1,i}+\xi_i\eta_{2,i}} u_{2,i}^{\theta_i} v_i^{\mu_i}\right) \times \prod_{k=1}^{t} \left(\prod_{i\in I_k} e_i^{a_{k,i}\lambda_k} A_k^{-\lambda_k}\right) = \prod_{i=1}^{n} \left(\mathsf{hp}_{1,i}\mathsf{hp}_{2,i}^{\xi_i}\right)^{r_i} \stackrel{\mathsf{def}}{=} H'.$$

Theorem 3. *The above* SPHF *is a type II* 0*-smooth (or perfectly smooth)* KV$-$SPHF.

4.3 CS$-$SPHF Construction

The hashing key hk is a random tuple $(\eta_1, \eta_2, \theta, \mu, \kappa, \lambda, \alpha) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^6 \times \{0, \ldots, 2^{\mathfrak{L}}-1\}$, where $\mathfrak{L}$ is a parameter (smoothness depends on $\mathfrak{L}$). We suppose $2^{\mathfrak{L}} < p$. The projection key is $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_3, \alpha)$ with:

$$\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} c^{\mu} \qquad \mathsf{hp}_2 = g_1^{\eta_2} d^{\mu} \qquad \mathsf{hp}_3 = g_1^{\kappa} h^{\lambda}.$$

The hash value is:

$$H \stackrel{\mathsf{def}}{=} \prod_{i=1}^{n} \left(u_{1,i}^{\eta_1+\xi_i\eta_2} u_{2,i}^{\theta} v_i^{\mu}\right)^{\alpha^{i-1}} \prod_{k=1}^{t} \left(\prod_{i\in I_k} \left(u_{1,i}^{a_{k,i}\kappa} e^{a_{k,i}\lambda}\right) A_k^{-\lambda}\right)^{\alpha^{k-1}} = \prod_{i=1}^{n} \left(\mathsf{hp}_1^{r_i}\mathsf{hp}_2^{\xi_i r_i}\right)^{\alpha^{i-1}} \prod_{k=1}^{t} \prod_{i\in I_k} \mathsf{hp}_3^{a_{k,i}r_i\alpha^{k-1}} \stackrel{\mathsf{def}}{=} H'.$$

Theorem 4. *The above* SPHF *is a type III* $m/2^{\mathfrak{L}-1}$*-smooth* CS$-$SPHF*, with* $m = \max(t, n) - 1$.

In the Appendix B.2, we show that this SPHF is not at all[1] a KV$-$SPHF.

4.4 GL$-$SPHF Construction

For this construction, we only consider a ciphertext of vector with a unique label ℓ, *i.e.*, all ξ_i are equal to $\xi = \mathfrak{H}_K(\ell, \mathbf{u}_1, \ldots, \mathbf{u}_n, e_1, \ldots, e_n)$. Intuitively, this construction consists in grouping hp_1 and hp_2 in the previous construction. This is possible because the ξ_i and so cd^{ξ_i} are the same for all commitments. Formally, the hashing key hk is a random tuple $(\eta, \theta, \mu, \kappa, \lambda, \alpha) \stackrel{\$}{\leftarrow} \mathbb{Z}_p^5 \times \{0, \ldots, 2^{\mathfrak{L}} - 1\}$. The projection key is $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_3, \mathfrak{L})$ with:

$$\mathsf{hp}_1 = g_1^{\eta} g_2^{\theta} (cd^{\xi})^{\mu} \qquad \mathsf{hp}_3 = g_1^{\kappa} h^{\lambda}.$$

[1] We mean that, for any $\varepsilon < 1$, it is not an ε-smooth KV$-$SPHF.

The hash value is:

$$H \stackrel{\text{def}}{=} \prod_{i=1}^{n} \left(u_{1,i}^{\eta} u_{2,i}^{\theta} v_i^{\mu}\right)^{\alpha^{i-1}} \times \prod_{k=1}^{t} \left(\prod_{i \in I_k} \left(u_{1,i}^{a_{k,i}\kappa} e^{a_{k,i}\lambda}\right) A_k^{-\lambda} \right)^{\alpha^{i-1}} = \prod_{i=1}^{n} \mathsf{hp}_1^{r_i \alpha^{i-1}} \times \prod_{k=1}^{t} \prod_{i \in I_k} \mathsf{hp}_{3,i}^{a_{k,i} r_i \alpha^{k-1}} \stackrel{\text{def}}{=} H'.$$

Theorem 5. *The above* SPHF *is a type III* $m/2^{\mathfrak{L}-1}$*-smooth* GL−SPHF*, with* $m = \max(t, n) - 1$.

4.5 Construction with Reuse of Randomness

In this section, we are interested in reusing commitment randomness to reduce the total size of both the commitments the projection keys. This will have a huge impact on LAKE with complex languages.

Commitment with Reuse of Randomness. We consider the following commitment scheme for vectors $(X_1, \ldots, X_n)$ of n elements in $\mathbb{G}$, which can be proven IND-CCA secure exactly as for the original Cramer-Shoup scheme [CS98]: from $4+n$ scalars $(x_1, x_2, y_1, y_2, z_1, \ldots, z_n) \stackrel{\$}{\leftarrow} {\mathbb{Z}_p}^{4+n}$, one sets $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, and $h_1 = g_1^{z_1}, \ldots, h_n = g_1^{z_n}$. The public parameters consist of $\mathsf{ek} = (\mathbb{G}, g_1, g_2, c, d, h_1, \ldots, h_n, \mathfrak{H}_K)$, while the trapdoor for extraction is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z_1, \ldots, z_n)$. One can define the encryption process, with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, \boldsymbol{e})$:

$$C \stackrel{\text{def}}{=} \mathsf{CS}(\ell, \mathsf{ek}, (X_1, \ldots, X_n); r) \stackrel{\text{def}}{=} (\mathbf{u} = (g_1^r, g_2^r), \boldsymbol{e} = (X_1 \cdot h_1^r, \ldots, X_n \cdot h_n^r), v = (cd^{\xi})^r).$$

KV−SPHF Construction. As for KV−SPHF for classical commitments, we propose two constructions: one of type III and one of type II, which are much more efficient.

Type III Scheme. The hashing key hk is a list of a scalar 4-tuple $\mathsf{hk}_1 = (\eta_1, \eta_2, \theta, \mu)$ and t scalar $(n+1)$-tuples $\mathsf{hk}_{3,k} = (\kappa_{k,1}, \ldots, \kappa_{k,n}, \lambda_k)$, for $k \in \{1, \ldots, t\}$. The projection key hp is composed of 2 projection sub-keys hp_1, hp_2 and tn projection sub-keys $\mathsf{hp}_{3,k,i}$ with:

$$\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} c^{\mu} \qquad \mathsf{hp}_2 = g_1^{\eta_2} d^{\mu} \qquad \mathsf{hp}_{3,k,i} = g_1^{\kappa_{k,i}} h_i^{\lambda_k}$$

The hash value is:

$$H \stackrel{\text{def}}{=} u_1^{\eta_1 + \xi\eta_2} u_2^{\theta} v^{\mu} \times \prod_{k=1}^{t} \left(\prod_{i \in I_k} \left(u_1^{a_{k,i}\kappa_{k,i}} e_i^{a_{k,i}\lambda_k}\right) A_k^{-\lambda_k} \right) = \left(\mathsf{hp}_1 \mathsf{hp}_2^{\xi} \times \prod_{k=1}^{t} \prod_{i \in I_k} \mathsf{hp}_{3,k,i}^{a_{k,i}} \right)^r \stackrel{\text{def}}{=} H'.$$

Theorem 6. *The above* SPHF *is a type III* 0*-smooth (or perfectly smooth)* KV−SPHF.

Type II Scheme. The hashing key hk is a list of a scalar 4-tuple $\mathsf{hk}_1 = (\eta_1, \eta_2, \theta, \mu)$ and t scalars $\mathsf{hk}_{3,k} = \lambda_k$, for $k \in \{1, \ldots, t\}$. The projection key hp is composed of 2 projections sub-keys hp_1 and hp_2 with:

$$\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} \left(\prod_{i=1}^{n} h_i^{\sum_{k=1}^{t} a_{k,i}\lambda_k} \right) c^{\mu} \qquad \mathsf{hp}_2 = g_1^{\eta_2} d^{\mu}$$

The hash value is:

$$H \stackrel{\text{def}}{=} \prod_{i=1}^{n} \left(u_1^{\eta_1 + \xi\eta_2} u_2^{\theta} v^{\mu}\right) \times \prod_{k=1}^{t} \left(\prod_{i \in I_k} e_i^{a_{k,i}} / A_k \right)^{\lambda_k} = (\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r \stackrel{\text{def}}{=} H'.$$

Theorem 7. *The above* SPHF *is a type II* 0*-smooth (or perfectly smooth)* KV−SPHF.

	EG		CS		Linear		Linear CS	
	hk	hp	hk	hp	hk	hp	hk	hp
KV III	$2t \times \mathbb{Z}_p$	$t \times \mathbb{G}$	$(4n+2t) \times \mathbb{Z}_p$	$(2n+t) \times \mathbb{G}$	$3t \times \mathbb{Z}_p$	$2t \times \mathbb{G}$	$(6n+3t) \times \mathbb{Z}_p$	$(4n+2t) \times \mathbb{G}$
KV II	$(n+t) \times \mathbb{Z}_p$	$n \times \mathbb{G}$	$(4n+t) \times \mathbb{Z}_p$	$2n \times \mathbb{G}$	$2(n+t) \times \mathbb{Z}_p$	$2n \times \mathbb{G}$	$(6n+2t) \times \mathbb{Z}_p$	$4n \times \mathbb{G}$
CS III	$2 \times \mathbb{Z}_p + \alpha$	$1 \times \mathbb{G} + \alpha$	$6 \times \mathbb{Z}_p + \alpha$	$3 \times \mathbb{G} + \alpha$	$3 \times \mathbb{Z}_p + \alpha$	$2 \times \mathbb{G} + \alpha$	$8 \times \mathbb{Z}_p + \alpha$	$6 \times \mathbb{G} + \alpha$
GL III	n/a	n/a	$5 \times \mathbb{Z}_p + \alpha$	$2 \times \mathbb{G} + \alpha$	n/a	n/a	$6 \times \mathbb{Z}_p + \alpha$	$4 \times \mathbb{G} + \alpha$

Table 1. Summary of our Constructions Without Reuse of Randomness (α corresponds to a number of size $\mathfrak{L}$)

	EG		CS		Linear		Linear CS	
	$\mathcal{C}$	hp	$\mathcal{C}$	hp	$\mathcal{C}$	hp	$\mathcal{C}$	hp
KV III	$2n \times \mathbb{G}$	$t \times \mathbb{G}$	$4n \times \mathbb{G}$	$(2n+t) \times \mathbb{G}$	$3n \times \mathbb{G}$	$2t \times \mathbb{G}$	$5n \times \mathbb{G}$	$(4n+2t) \times \mathbb{G}$
KV III rr	$(n+1) \times \mathbb{G}$	$tn \times \mathbb{G}$	$(n+3) \times \mathbb{G}$	$(tn+2) \times \mathbb{G}$	$(n+2) \times \mathbb{G}$	$2tn \times \mathbb{G}$	$(n+4) \times \mathbb{G}$	$(2tn+4) \times \mathbb{G}$
KV II	$2n \times \mathbb{G}$	$n \times \mathbb{G}$	$4n \times \mathbb{G}$	$2n \times \mathbb{G}$	$3n \times \mathbb{G}$	$2n \times \mathbb{G}$	$5n \times \mathbb{G}$	$4n \times \mathbb{G}$
KV II rr	$(n+1) \times \mathbb{G}$	$1 \times \mathbb{G}$	$(n+3) \times \mathbb{G}$	$2 \times \mathbb{G}$	$(n+2) \times \mathbb{G}$	$2 \times \mathbb{G}$	$(n+4) \times \mathbb{G}$	$4 \times \mathbb{G}$
CS III	$2n \times \mathbb{G}$	$1 \times \mathbb{G} + \alpha$	$4n \times \mathbb{G}$	$3 \times \mathbb{G} + \alpha$	$3n \times \mathbb{G}$	$2 \times \mathbb{G} + \alpha$	$5n \times \mathbb{G}$	$6 \times \mathbb{G} + \alpha$
GL III	n/a	n/a	$4n \times \mathbb{G}$	$2 \times \mathbb{G} + \alpha$	n/a	n/a	$5n \times \mathbb{G}$	$4 \times \mathbb{G} + \alpha$
GL II rr	n/a	n/a	$(n+3) \times \mathbb{G}$	$1 \times \mathbb{G}$	n/a	n/a	$(n+4) \times \mathbb{G}$	$2 \times \mathbb{G}$

Table 2. Summary of our Constructions (constructions with reuse of randomness are suffixed by "rr" and α corresponds to a number of size $\mathfrak{L}$)

GL−SPHF Construction. This construction consists in mixing hp_1 and hp_2 from the above KV−SPHF construction. The hashing key hk is a list of a scalar triple $\mathsf{hk}_1 = (\eta, \theta, \mu)$ and t scalar n-tuples $\mathsf{hk}_{3,k} = (\lambda_{k,1}, \ldots, \lambda_{k,n})$, for $k \in \{1, \ldots, t\}$. The projection key hp and the hash values are:

$$\mathsf{hp} = g_1^{\eta} g_2^{\theta} \left(\prod_{i=1}^{n} h_i^{\sum_{k=1}^{t} a_{k,i} \lambda_k} \right) \left(cd^{\xi} \right)^{\mu} \qquad H \stackrel{\mathsf{def}}{=} \prod_{i=1}^{n} \left(u_1^{\eta} u_2^{\theta} v^{\mu} \right) \times \prod_{k=1}^{t} \left(\prod_{i \in I_k} e_i^{a_{k,i}} / A_k \right)^{\lambda_k} = \mathsf{hp}_1^r \stackrel{\mathsf{def}}{=} H'.$$

Theorem 8. *The above* SPHF *is a type II* 0*-smooth (or perfectly smooth)* GL−SPHF.

4.6 Extensions and Summary

It is possible to use the same ideas as in the Appendix B.1, to extend all the previous constructions to ElGamal (EG), Linear and Linear CS commitment schemes. The costs of all the constructions can be found in Tables 1 and 2.

5 Applications to LAKE

In this section, we show how our new KV−SPHF enable us to design efficient Language-Authenticated Key Exchange (LAKE), in the standard model. We insist we do not try to make them in the UC-framework as [BCPV12], but efficiency is our priority: we achieve that in one-round only.

LAKE is a general framework proposed by the authors of [BCPV12]: It generalizes several AKE primitives such as PAKE, Verifier-based PAKE or secret handshakes. In such a scheme, each player U owns a word W in a certain language L and expect the other player to own a word W' in another language L$'$. If everything is compatible (ie 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 values sent by the other player. 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 each commit on the private part priv for the language L they want to use, the private part $priv$ for the language L$'$ they assume the other player will use, and the word W they own in their language L.

5.1 Security Model

For the security model, we use the Bellare-Rogaway [BR94] model, with some additions from the 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.

The quality of an adversary is measured by its ability to distinguish the answer of the Test-query on a fresh instance. Its advantage should be negligible, or negligibly close to q_s/N when the authentication means can be guessed with probability $1/N$, where q_s is the number of Send-queries which model active attacks. The latter case covers on-line dictionary attacks. More details can be found in [BR94, BPR00].

5.2 Instantiation

Using the same approach as our previous PAKE scheme, one can design the scheme proposed on Figure 2, in which both users U and U' have agreed on the public part pub, while U owns a word W for a private part priv, and thinks to a private part $priv'$ for U', and while U' owns a word W' for a private part priv$'$, and thinks to a private part $priv$ for U.

U will thus define $\mathsf{aux} = (\mathsf{pub}, \mathsf{priv}, priv')$ and $aux = (\mathsf{pub}, priv', \mathsf{priv})$ and will commit $(\mathsf{priv}, priv', W)$ into C with random coins r. The ciphertext C of the triple (a, b, c) will be in $\text{LOFC}_{\mathsf{aux}}$ if $a = \mathsf{priv}$, $b = priv'$, and $c \in \mathrm{L}_{\mathsf{aux}}$ (where $priv'$ can be ignored). U' will do the same, defining $\mathsf{aux}' = (\mathsf{pub}, \mathsf{priv}', priv)$ and $aux' = (\mathsf{pub}, priv, \mathsf{priv}')$, and committing $(\mathsf{priv}', priv, W')$ into C' with random coins r'. The ciphertext C' of the triple (a, b, c) will be in $\text{LOFC}_{\mathsf{aux}'}$ if $a = \mathsf{priv}'$, $b = priv$, and $c \in \mathrm{L}_{\mathsf{aux}'}$. Using a similar proof as the one in [KV11], one can state (more details can be found in the Appendix C):

Theorem 9. *If the encryption scheme is* $\mathsf{IND-CCA}$*, and* $\text{LOFC}_{\mathsf{aux}}$ *languages admit type II* $\mathsf{KV-SPHF}$*, then our* LAKE *protocol is secure: the advantage of any adversary is bounded by* $q_s \times 2^{-m} \times \mathsf{Succ}^{\mathrm{L}}(t) + \mathsf{negl}()$*, where* q_s *is the number of* Send*-queries, and* m *is the minimal min-entropy of the pairs* $(\mathsf{priv}, \mathsf{priv}')$ *for any two users* U*,* U'*, and* $\mathsf{Succ}^{\mathrm{L}}(t)$ *is the best success an adversary can get in finding a word in a language* $\mathrm{L}_{\mathsf{aux}}$ *of any player, within time* t*.*

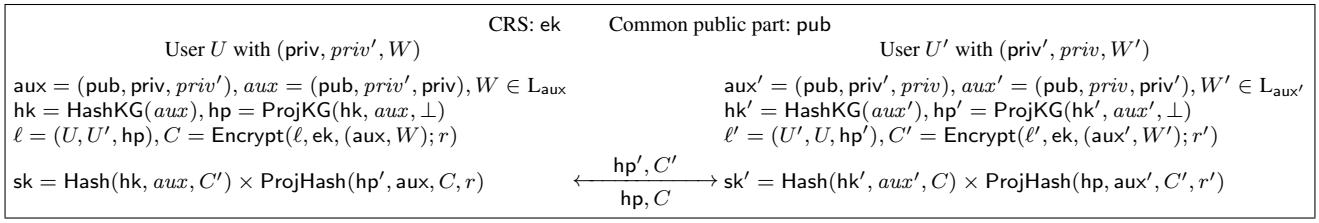

Fig. 2. One-Round LAKE

One can see that this result is optimal in the sense that if we consider the PAKE or Verifier-based PAKE, m is the min-entropy of the dictionary, and once the password is found, L is a trivial language: $2^{-m} = 1/N$ and $\mathsf{Succ}^{\mathrm{L}}(t) = 1$, hence the bound is q_s/N, where N is the size of the dictionary, if one assumes passwords are uniformly chosen.

If we consider secret handshakes, the values priv specify the verification keys corresponding to the private keys under which messages are signed. The entropy of the verification keys is small, since the number of possible authorities is low, but in this case, $\mathsf{Succ}^{\mathrm{L}}(t)$ is the success probability to forge a signature, which is negligible.

With type III languages, one could precompute the hashing and projection keys before the execution of the protocol.

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.

[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.

[BCPV12] Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient uc-secure authenticated key-exchange for algebraic languages. Cryptology ePrint Archive, Report 2012/284, 2012. `http://eprint.iacr.org/`.

[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.

[BR94] Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In Douglas R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer, August 1994.

[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.

[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.

[CKP07] Ronald Cramer, Eike Kiltz, and Carles Padró. A note on secure computation of the Moore-Penrose pseudoinverse and its application to secure linear algebra. In Alfred Menezes, editor, *Advances in Cryptology – CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Science*, pages 613–630. Springer, August 2007.

[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.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, August 1985.

[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.

[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.

[Sha07] Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074, 2007. http://eprint.iacr.org/2007/074.pdf.

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 $\{0,1\}^k$ 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

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

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 $\perp$.

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 $\mathcal{CT}$.

$\mathsf{Exp}^{\mathsf{ind-cca}-b}_{\mathcal{E},\mathcal{A}}(k)$
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'

The advantages are

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

Labeled commitment scheme. A labeled commitment scheme is defined by three algorithms:

- $\mathsf{Setup}(1^{\mathfrak{K}})$, where $\mathfrak{K}$ is the security parameter, generates the global parameters param of the scheme;
- $\mathsf{Commit}(\ell, m; r)$ produces a commitment c and the opening information d on the input message $m \in \mathcal{M}$ under the label ℓ, using the random coins r;
- $\mathsf{Decommit}(\ell, c, m, d)$ checks the validity of the opening information d on the commitment c for the message m under the label ℓ. It answers 1 for true, and 0 for false.

A commitment scheme $\mathcal{C}$ should satisfy the following properties

- *Correctness*: for any label ℓ, and all messages m, if $(c, d) \leftarrow \mathsf{Commit}(\ell, m; r)$, then $\mathsf{Decommit}(\ell, c, m, d) = 1$.
- *Hiding*: this security notion is similar to the indistinguishability under chosen-plaintext attacks for encryption, which means that c does not help to distinguish between to candidates m_0 and m_1 as committed values.
- *Binding*: this security notion is more an unforgeability notion, which means that for any commitment c, it should be hard to open it in two different ways, which means to exhibit (m_0, d_0) and (m_1, d_1), such that $m_0 \neq m_1$ and $\mathsf{Decommit}(\ell, c, m_0, d_0) = \mathsf{Decommit}(\ell, c, m_1, d_1) = 1$.

The commitment algorithm can be interactive between the sender and the received, but the hiding and the binding properties should still hold. Several additional properties are sometimes required:

- *Extractability*: an indistinguishable Setup procedure also outputs a trapdoor that allows a extractor to get the committed value m from any commitment c. More precisely, if c can be open in a valid way, the extractor can get this value from the commitment.
- *Equivocability*: an indistinguishable Setup procedure also outputs a trapdoor that allows a simulator to generate commitments that can thereafter be open in any way.
- *Non-Malleability*: it should be hard, from a commitment c to generate a new commitment $c' \neq c$ whose committed values are in relation.

A.2 Computational Assumptions

The two classical decisional assumptions we use along this paper are: the decisional Diffie-Hellman (DDH) and the decisional Linear (DLin) assumptions. Our constructions essentially rely on the DDH assumption, that implies the DLin assumption.

Definition 10 (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.*

Definition 11 (Decisional Linear Problem (DLin)). *The Decisional Linear Problem [BBS04] says that, in a group $(p, \mathbb{G}, g)$, when we are given $(g^x, g^y, g^{xa}, g^{yb}, g^c)$ for unknown random $x, y, a, b \xleftarrow{\$} \mathbb{Z}_p$, it is hard to decide whether $c = a+b \bmod p$ (a linear tuple) or $c \xleftarrow{\$} \mathbb{Z}_p$ (a random tuple). We define by $\mathsf{Adv}^{\mathsf{dlin}}_{p,\mathbb{G},g}(t)$ the best advantage an adversary can have in distinguishing a linear tuple from a random tuple within time t.*

A.3 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:

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

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.4 Some Encryption Schemes

Cramer-Shoup (CS) Encryption Scheme.
The Cramer-Shoup encryption scheme [CS98] can be tuned 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 if one uses a collision-resistant hash function $\mathcal{H}$.

Linear Cramer-Shoup (LCS) Encryption Scheme.
The Linear Cramer-Shoup encryption scheme [CKP07,Sha07] can be tuned to a labeled public-key encryption scheme:

- $\mathsf{Setup}(1^{\mathfrak{K}})$ generates a group $\mathbb{G}$ of order p, with three independent generators $(g_1, g_2, g_3) \xleftarrow{\$} \mathbb{G}^3$;
- $\mathsf{KeyGen}(\mathsf{param})$ generates $\mathsf{dk} = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \xleftarrow{\$} \mathbb{Z}_p^9$, and sets, for $i \in \{1, 2\}$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$. It also chooses a hash function $\mathfrak{H}_K$ in a collision-resistant hash family $\mathcal{H}$ (or simply a Universal One-Way Hash Function). The encryption key is $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$.
- $\mathsf{Encrypt}(\ell, \mathsf{ek}, M; r, s)$, for a message $M \in \mathbb{G}$ and two random scalars $r, s \xleftarrow{\$} \mathbb{Z}_p$, the ciphertext is $C = (\ell, \mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^{\xi})^r (c_2 d_2^{\xi})^s)$, where v is computed afterwards with $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$.
- $\mathsf{Decrypt}(\ell, \mathsf{dk}, C = (\mathbf{u}, e, v))$: one first computes $\xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$ and checks whether $u_1^{x_1 + \xi y_1} \cdot u_2^{x_2 + \xi y_2} \cdot u_3^{x_3 + \xi y_3} \stackrel{?}{=} v$. If the equality holds, one computes $M = e / (u_1^{z_1} u_2^{z_2} u_3^{z_3})$ and outputs M. Otherwise, one outputs $\perp$.

This scheme is indistinguishable against chosen-ciphertext attacks, under the DLin assumption and if one uses a collision-resistant hash function $\mathcal{H}$.

B Smooth Projective Hash Functions

In this appendix, we first present variants of the KV−SPHF described in Section 2.4, for other commitment schemes, such as ElGamal and DLin-based commitments scheme. We then prove all the theorems of Section 4, concerning SPHF for more complex equations. Finally, we show how to deal with systems of linear and quadratic pairing product equations.

B.1 KV−SPHF for Constant Committed Value or Valid Commitments

All constructions of this section are very similar to the construction of Section 2.4. Therefore, we do not provide any proof.

DDH-Based Constructions

SPHF *on ElGamal Ciphertexts.* The ElGamal encryption scheme [ElG85] is defined on a group $\mathbb{G}$ of order p, with a generator g. The decryption key dk is a scalar $x \in \mathbb{Z}_p$, and the encryption key is $\mathsf{ek} = h = g^x$:

$$C = \mathsf{Encrypt}(\mathsf{ek}, M; r) = (c_1 = g^r, c_2 = M \cdot h^r) \quad \mathsf{Decrypt}(\mathsf{dk}, C) = c_2/c_1^x.$$

For the SPHF on encryptions of M, one generates $\mathsf{hk} = (\eta, \lambda) \xleftarrow{\$} \mathbb{Z}_p^2$, and derives the projection key $\mathsf{hp} = g^\eta h^\lambda$. Then, one can compute the hash value:

$$H = \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\mathsf{def}}{=} c_1^\eta (c_2/M)^\lambda = \mathsf{hp}^r \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C; r) = H'.$$

SPHF *on Cramer-Shoup Ciphertexts.* The labeled Cramer-Shoup encryption scheme [CS98] is defined on a group $\mathbb{G}$ of prime order p, with two generators $(g_1, g_2) \xleftarrow{\$} \mathbb{G}^2$. The decryption key is $\mathsf{dk} = (x_1, x_2, y_1, y_2, z) \xleftarrow{\$} \mathbb{Z}_p^5$, and the encryption key is $\mathsf{ek} = (g_1, g_2, c, d, h, \mathfrak{H}_K)$, where $c = g_1^{x_1} g_2^{x_2}$, $d = g_1^{y_1} g_2^{y_2}$, $h = g_1^z$, and $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$ (in a collision-resistant hash family).

$$C = \mathsf{Encrypt}(\ell, \mathsf{ek}, M; r) = (\ell, \mathbf{u} = (g_1^r, g_2^r), e = M \cdot h^r, v = (cd^\xi)^r) \text{ where } \xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$$
$$\mathsf{Decrypt}(\ell, \mathsf{dk}, C) = e/u_1^z \text{ if } u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} = v; \perp \text{ otherwise.}$$

For the SPHF on encryptions of M, one generates $\mathsf{hk} = (\eta_1, \eta_2, \theta, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, and derives the projection key $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta h^\lambda c^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d^\mu)$. Then one can compute the hash value:

$$H \stackrel{\mathsf{def}}{=} \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\mathsf{def}}{=} u_1^{(\eta_1+\xi\eta_2)} u_2^\theta (e/M)^\lambda v^\mu = (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, r) \stackrel{\mathsf{def}}{=} H'.$$

For the SPHF on valid Cramer-Shoup Ciphertexts, one generates $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu) \xleftarrow{\$} \mathbb{Z}_p^4$, and derives the projection key $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^\theta c^\mu, \mathsf{hp}_2 = g_1^{\eta_2} d^\mu)$. Then one can compute the hash value:

$$H \stackrel{\mathsf{def}}{=} \mathsf{Hash}(\mathsf{hk}, C) \stackrel{\mathsf{def}}{=} u_1^{(\eta_1+\xi\eta_2)} u_2^\theta v^\mu = (\mathsf{hp}_1 \mathsf{hp}_2^\xi)^r \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, C, r) \stackrel{\mathsf{def}}{=} H'.$$

DLin-Based Constructions

SPHF *on Linear Ciphertexts* The Linear encryption scheme [BBS04] is defined on a group $\mathbb{G}$ of prime order p, with a generator $h \xleftarrow{\$} \mathbb{G}$. The decryption key is $\mathsf{dk} = (z_1, z_2) \xleftarrow{\$} \mathbb{Z}_p^2$, and the encryption key is $\mathsf{ek} = (g_1 = h^{z_1}, g_2 = h^{z_2}, h)$:

$$C = \mathsf{Encrypt}(\mathsf{ek}, M; r, s) = (c_1 = g_1^r, c_2 = g_2^s, c_3 = M \cdot h^{r+s}) \quad \mathsf{Decrypt}(\mathsf{dk}, C) = c_3/(c_1^{1/z_1} c_2^{1/z_2}).$$

For the SPHF on encryptions of M, one generates $\mathsf{hk} = (\eta, \theta, \lambda) \xleftarrow{\$} \mathbb{Z}_p^3$, and derives the projection key $\mathsf{hp} = (\mathsf{hp}_1 = g_1^\eta h^\lambda, \mathsf{hp}_2 = g_2^\theta h^\lambda)$. Then, one can compute the hash value:

$$H = \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\mathsf{def}}{=} c_1^\eta c_2^\theta (c_3/M)^\lambda = \mathsf{hp}_1^r \mathsf{hp}_2^s \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, (r, s)) = H'.$$

SPHF *on Linear Cramer-Shoup Ciphertexts.* The Linear Cramer-Shoup encryption scheme [CKP07, Sha07] is defined on a group $\mathbb{G}$ of prime order p, with three independent generators $(g_1, g_2, g_3) \xleftarrow{\$} \mathbb{G}^3$. The decryption key is $\mathsf{dk} = (x_1, x_2, x_3, y_1, y_2, y_3, z_1, z_2, z_3) \xleftarrow{\$} \mathbb{Z}_p^9$, and the encryption key is $\mathsf{ek} = (c_1, c_2, d_1, d_2, h_1, h_2, \mathfrak{H}_K)$, where, for $i \in \{1, 2\}$, $c_i = g_i^{x_i} g_3^{x_3}$, $d_i = g_i^{y_i} g_3^{y_3}$, and $h_i = g_i^{z_i} g_3^{z_3}$, and $\mathfrak{H}_K \xleftarrow{\$} \mathcal{H}$ (in a collision-resistant hash family).

$$C = \mathsf{Encrypt}(\ell, \mathsf{ek}, M; r, s) = (\ell, \mathbf{u} = (g_1^r, g_2^s, g_3^{r+s}), e = M \cdot h_1^r h_2^s, v = (c_1 d_1^\xi)^r (c_2 d_2^\xi)^s), \text{ where } \xi = \mathfrak{H}_K(\ell, \mathbf{u}, e)$$
$$\mathsf{Decrypt}(\ell, \mathsf{dk}, C) = e/(u_1^{z_1} u_2^{z_2} u_3^{z_3}) \text{ if } u_1^{x_1+\xi y_1} \cdot u_2^{x_2+\xi y_2} \cdot u_3^{x_3+\xi y_3} = v; \perp \text{ otherwise.}$$

For the SPHF on encryptions of M, one generates $\mathsf{hk} = (\eta_1, \eta_2, \theta_1, \theta_2, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^7$, and derives the projection key

$$\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_3^{\kappa} h_1^{\lambda} c_1^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d_1^{\mu}, \mathsf{hp}_3 = g_2^{\theta_1} g_3^{\kappa} h_2^{\lambda} c_2^{\mu}, \mathsf{hp}_4 = g_2^{\theta_2} d_2^{\mu}).$$

Then, one can compute the hash value (when $\xi \neq 0$):

$$H = \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\mathsf{def}}{=} u_1^{\eta_1+\xi\eta_2} u_2^{\theta_1+\xi\theta_2} u_3^{\kappa} (e/M)^{\lambda} v^{\mu} = (\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r (\mathsf{hp}_3 \mathsf{hp}_4^{\xi})^s \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, (r, s)) = H'.$$

For the SPHF on valid Linear Cramer-Shoup Ciphertexts, one generates $\mathsf{hk} = (\eta_1, \eta_2, \theta_1, \theta_2, \kappa, \mu) \xleftarrow{\$} \mathbb{Z}_p^6$, and derives the projection key

$$\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_3^{\kappa} c_1^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d_1^{\mu}, \mathsf{hp}_3 = g_2^{\theta_1} g_3^{\kappa} c_2^{\mu}, \mathsf{hp}_4 = g_2^{\theta_2} d_2^{\mu}).$$

Then, one can compute the hash value (when $\xi \neq 0$):

$$H = \mathsf{Hash}(\mathsf{hk}, C) \stackrel{\mathsf{def}}{=} u_1^{\eta_1+\xi\eta_2} u_2^{\theta_1+\xi\theta_2} u_3^{\kappa} v^{\mu} = (\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r (\mathsf{hp}_3 \mathsf{hp}_4^{\xi})^s \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, C, (r, s)) = H'.$$

DLin-Based Constructions with Pairings. In a pairing-friendly environment one can see that a Linear encryption in a group $\mathbb{G}$ induces an ElGamal encryption in the target group $\mathbb{G}_T$. In other word testing the *Linear*-validity of a tuple (g_1^r, g_2^s, h^{r+s}) is equivalent to the *DH*-validity of the pair $(e(g_1^r, g_2) \cdot e(g_1, g_2^s), e(h^{r+s}, g_2)) = (e(g_1, g_2)^{r+s}, e(h, g_2)^{r+s})$ in basis $(e(g_1, g_2), e(h, g_2))$. Following this idea, one can simplify the previous projection keys by only building them for the witness r, or alternatively $t = r + s$, and if the witness is correct then a certain pairing equation should hold.

SPHF *on Linear Ciphertexts.* For the revised SPHF on encryptions of M, one generates $\mathsf{hk} = (\eta, \lambda) \xleftarrow{\$} \mathbb{Z}_p^2$, and derives the projection key $\mathsf{hp} = g_1^{\eta} h^{\lambda}$. Then, one can compute the hash value:

$$H = \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\mathsf{def}}{=} (e(c_1, g_2) \cdot e(c_2, g_1))^{\eta} \cdot e(c_3/M, g_2)^{\lambda} = e(\mathsf{hp}^t, g_2) \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, t) = H'.$$

SPHF *on Linear Cramer-Shoup Ciphertexts.* For the revised SPHF on encryptions of M, one generates $\mathsf{hk} = (\eta_1, \eta_2, \kappa, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$, and derives the projection key

$$\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_3^{\kappa} h_1^{\lambda} c_1^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d_1^{\mu}).$$

Then, one can compute the hash value (when $\xi \neq 0$):

$$H = \mathsf{Hash}(\mathsf{hk}, M, C) \stackrel{\mathsf{def}}{=} e(u_1, g_2)^{\eta_1+\xi\eta_2} \cdot (e(u_3, g_2)/e(u_2, g_3))^{\kappa} \cdot (e(e/M, g_2)/e(h_2, u_2))^{\lambda} \cdot \Big(e(v, g_2)/e(c_2 d_2^{\xi}, u_3)\Big)^{\mu}$$
$$= e((\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r, g_2) \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, M, C, r) = H'.$$

For the SPHF on valid Linear Cramer-Shoup Ciphertexts, one generates $\mathsf{hk} = (\eta, \eta', \kappa, \mu) \xleftarrow{\$} \mathbb{Z}_p^4$, and derives the projection key

$$\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_3^{\kappa} c_1^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d_1^{\mu}).$$

Then, one can compute the hash value (when $\xi \neq 0$):

$$H = \mathsf{Hash}(\mathsf{hk}, C) \stackrel{\mathsf{def}}{=} e(u_1, g_2)^{\eta_1+\xi\eta_2} \cdot (e(u_3, g_2)/e(u_2, g_3))^{\kappa} \cdot \Big(e(v, g_2)/e(c_2 d_2^{\xi}, u_3)\Big)^{\mu}$$
$$= e((\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r, g_2) \stackrel{\mathsf{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, C, r) = H'.$$

B.2 Proofs of Theorems of Section 4

KV−SPHF Construction (Type III)

Proof (of Theorem 2). Let us write C_i as:

$$C_i = (\ell_i, \mathbf{u}_i = (g_1^{r_i}, g_2^{s_i}), e_i = X_i \cdot h^{r_i}, v_i = c^{r_i} d^{\xi_i t_i}),$$

with $X_i \in \mathbb{G}$ and $r_i, s_i, t_i \in \mathbb{Z}_p$. We want to prove that, when one commitment C_i is invalid ($s_i \neq r_i$ or $t_i \neq r_i$), or when one of the equations (1.1), ..., (1.t) is not satisfied, hp gives no information on the value of H. Let us write $g_2 = g_1^{\beta}$, $c = g_1^{\gamma}$, and $d = g_1^{\delta}$. We suppose $\beta \neq 0$, $\gamma \neq 0$, $\delta \neq 0$ and $\xi_i \neq 0$, for all $i = 1, \ldots, n$. As in the proof of Theorem 1, H is unpredictable given hp, if and only if the last line of one of the following matrices is linearly independent from the other lines of this matrix:

$$\begin{pmatrix} 1 & 0 & \beta & \gamma \\ 0 & 1 & 0 & \delta \\ r_i & \xi_i r_i & s_i\beta & r_i\gamma + \xi_i t_i \delta \end{pmatrix} \qquad \text{for } i \in \{1, \ldots, n\} \text{ — columns: } \eta_{1,i}, \eta_{2,i}, \theta_i, \mu_i$$

$$\begin{pmatrix} 1 & z \\ \sum_{i \in I_k} a_{k,i} r_i & \sum_{i \in I_k} a_{k,i} r_i z + \Delta_k \end{pmatrix} \qquad \text{for } k \in \{1, \ldots, t\} \text{ — columns: } \kappa_k, \lambda_k$$

with $\Delta_k = \log_{g_1}\left((\prod_{i \in I_k} X_i^{a_{k,i}})/A_k\right)$, for $k \in \{1, \ldots, t\}$. This is the case if and only if $s_i = t_i = r_i$, for $i \in \{1, \ldots, n\}$, and $\Delta_k = 0$, for $k \in \{1, \ldots, t\}$, *i.e.*, if all commitments are valid and all equations are verified. □

KV−SPHF Construction (Type II)

Proof (of Theorem 3). The proof for the smoothness is almost the same as the proof of Theorem 2, except the second series of matrices:

$$\begin{pmatrix} 1 & z \\ \sum_{i \in I_k} a_{k,i} r_i & \sum_{i \in I_k} a_{k,i} r_i z + \Delta_k \end{pmatrix} \qquad \text{for } k \in \{1, \ldots, t\} \text{ — columns: } \kappa_k, \lambda_k$$

is replaced by

$$\begin{pmatrix} 1 & 0 & 0 & a_{k,1} z \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & a_{k,t} z \\ r_1 & \ldots & r_n & \sum_{i \in I_k} a_{k,i} r_i z + \Delta_k \end{pmatrix} \qquad \text{for } k \in \{1, \ldots, t\} \text{ — columns: } \eta_{1,1}, \ldots, \eta_{1,n}, \lambda_k.$$

Finally, the scheme is (perfectly or 0-indistinguishable) type II, because it is clear that given hp, all $a_{k,i}$'s are completely unpredictable. □

CS−SPHF Construction

Proof (of Theorem 4). We use the same notations as in the proof of Theorem 2. H is unpredictable given hp, if and only if the last line of one of the following matrices is linearly independent from the other lines of this matrix:

$$\begin{pmatrix} 1 & 0 & \beta & \gamma \\ 0 & 1 & 0 & \delta \\ \sum_{i=1}^{n} r_i \alpha^{i-1} & \sum_{i=1}^{n} \xi_i r_i \alpha^{i-1} & \sum_{i=1}^{n} s_i \beta \alpha^{i-1} & \sum_{i=1}^{n} (r_i\gamma + \xi_i t_i \delta) \alpha^{i-1} \end{pmatrix} \qquad \text{columns: } \eta_1, \eta_2, \theta, \mu$$

$$\begin{pmatrix} 1 & z \\ \sum_{k=1}^{t} \left(\sum_{i \in I_k} a_{k,i} r_i\right) \alpha^{k-1} & \sum_{k=1}^{t} \left(\sum_{i \in I_k} a_{k,i} r_i z + \Delta_k\right) \alpha^{k-1} \end{pmatrix} \qquad \text{columns: } \kappa, \lambda$$

with $\Delta_k = \log_{g_1}\left(\prod_{i\in I_k} X_i^{a_{k,i}}/A_k\right)$, for $k \in \{1,\ldots,t\}$. This is the case if and only if $\sum_{i=1}^n (s_i - r_i)\alpha^{i-1} = 0$, $\sum_{i=1}^n (t_i - r_i)\alpha^{i-1} = 0$ and $\sum_{k=1}^t \Delta_k \alpha^{k-1} = 0$. These are three polynomials of unknown α in $\mathbb{Z}_p$ and of degree at most $m = \max(t,n) - 1$. Therefore, if one the coefficients of one polynomial is not null, there are at most m values of α such that the evaluation of this polynomial in α is 0. Furthermore, all commitments are valid and all equations are satisfied if and only if all coefficients of all polynomials are null. Since there are $2^{\mathfrak{L}}$ possible values of α, if any commitment is invalid or any equation is not verified, with probability at least $1 - m/2^{\mathfrak{L}}$, H will be unpredictable. Therefore, the above SPHF is ε-smooth with:

$$\varepsilon \leq \sum_{\alpha=0}^{m-1} \frac{1}{2^{\mathfrak{L}}} + \sum_{\alpha=m}^{2^{\mathfrak{L}}-1} \left(\frac{1}{2^{\mathfrak{L}}-m} - \frac{1}{2^{\mathfrak{L}}}\right) = \frac{m}{2^{\mathfrak{L}}} + \frac{2^{\mathfrak{L}}-m}{2^{\mathfrak{L}}-m} - \frac{2^{\mathfrak{L}}-m}{2^{\mathfrak{L}}} = 2 \times \frac{m}{2^{\mathfrak{L}}}$$

(supposing, wlog., that the bad values for α are $0,\ldots,m-1$). □

Proof (of the fact that this CS−SPHF *is not a* KV−SPHF*).* In the smoothness definition for a KV−SPHF, the adversary can choose C_i after having seen hp, and in particular, after knowing α. Therefore, since it is unbounded, it can create invalid $C_i = (\ell_i, \mathbf{u}_i = (g_1^{r_i}, g_2^{s_i}), e = h^{r_i}\cdot X_i, v = c^{r_i} d^{\xi_i t_i})$ for $i \in \{1,\ldots,n\}$ still verifying $\sum_{i=1}^n (s_i - r_i)\alpha^{i-1} = 0$, $\sum_{i=1}^n (t_i - r_i)\alpha^{i-1} = 0$ and $\sum_{k=1}^t \Delta_k \alpha^{k-1} = 0$ with $\Delta_k = \log_{g_1}\left(\prod_{i\in I_k} X_k/A_k\right)$. □

GL−SPHF Construction

Proof (of Theorem 5). Let us write C_i as:

$$C_i = (\ell_i, \mathbf{u}_i = (g_1^{r_i}, g_2^{s_i}), e_i = X_i \cdot h^{r_i}, v_i = c^{r_i} d^{\xi t_i}),$$

with $X_i \in \mathbb{G}$ and $r_i, s_i, t_i \in \mathbb{Z}_p$ and $\xi = \mathfrak{H}_K(\ell_1,\ldots,\ell_n,\mathbf{u}_1,\ldots,\mathbf{u}_n,e_1,\ldots,e_n)$.

The proof is nearly the same as the proof of Theorem 4. The only difference is that the matrix:

$$\begin{pmatrix} 1 & 0 & \beta & \gamma \\ 0 & 1 & 0 & \delta \\ \sum_{i=1}^n r_i\alpha^{i-1} & \sum_{i=1}^n \xi_i r_i \alpha^{i-1} & \sum_{i=1}^n s_i\beta\alpha^{i-1} & \sum_{i=1}^n (r_i\gamma + \xi_i t_i \delta)\alpha^{i-1} \end{pmatrix} \qquad \text{columns: } \eta_1, \eta_2, \theta, \mu$$

is replaced by the matrix:

$$\begin{pmatrix} 1 & \beta & \gamma + \xi\delta \\ \sum_{i=1}^n r_i\alpha^{i-1} & \sum_{i=1}^n s_i\beta\alpha^{i-1} & \sum_{i=1}^n (r_i\gamma + \xi t_i \delta)\alpha^{i-1} \end{pmatrix} \qquad \text{columns: } \eta, \theta, \mu.$$

□

Reuse of Randomness

KV−SPHF *Construction (Type III)*

Proof (of Theorem 6). We use the same notations as in the proof of Theorem 2. H is unpredictable given hp, if and only if the last line of one of the following matrices is linearly independent from the other lines of this matrix:

$$\begin{pmatrix} 1 & 0 & \beta & \gamma \\ 0 & 1 & 0 & \delta \\ r & \xi r & s & r\gamma + \xi t\delta \end{pmatrix} \qquad \text{columns: } \eta_1, \eta_2, \theta, \mu$$

$$\begin{pmatrix} 1 & 0 & 0 & z_1 \\ 0 & \ddots & 0 & \vdots \\ 0 & 0 & 1 & z_n \\ a_{k,1}r & \ldots & a_{k,n}r & \sum_{i\in I_k} a_{k,i} r z_i + \Delta_k \end{pmatrix} \qquad \text{for } k \in \{1,\ldots,t\} \text{ — columns: } \kappa_{k,1},\ldots,\kappa_{k,n},\lambda_k.$$

with $\Delta_k = \log_{g_1}\left(\prod_{i\in I_k} X_i^{a_{k,i}}/A_k\right)$, for $k \in \{1,\ldots,t\}$. This is the case if and only if $s = t = r$ and $\Delta_k = 0$, for $k \in \{1,\ldots,t\}$. Therefore, the SPHF is perfectly smooth. □

KV−SPHF *Construction (Type II)*

Proof (of Theorem 7). We use the same notations as in the proof of Theorem 2. H is unpredictable given hp, if and only if the last line of one of the following matrices is linearly independent from the other lines of this matrix:

$$\begin{pmatrix} 1 & 0 & \beta & \sum_{i\in I_k} a_{k,i} z_i & \gamma \\ 0 & 1 & 0 & 0 & \delta \\ r & \xi r & s & \sum_{i\in I_k} a_{k,i} r z_i + \Delta_k & r\gamma + \xi t\delta \end{pmatrix} \qquad \text{for } k \in \{1,\ldots,t\} \text{ — columns: } \eta_1, \eta_2, \theta, \lambda_k, \mu.$$

with $\Delta_k = \log_{g_1}\left(\prod_{i\in I_k} X_i^{a_{k,i}}/A_k\right)$, for $k \in \{1,\ldots,t\}$. This is the case if and only if $s = t = r$ and $\Delta_k = 0$, for $k \in \{1,\ldots,t\}$. Therefore, the SPHF is perfectly smooth. □

GL−SPHF *Construction*

Proof (of Theorem 8). The proof is nearly the same as the proof of Theorem 6. The only difference is that the matrices:

$$\begin{pmatrix} 1 & 0 & \beta & \sum_{i\in I_k} a_{k,i} z_i & \gamma \\ 0 & 1 & 0 & 0 & \delta \\ r & \xi r & s & \sum_{i\in I_k} a_{k,i} r z_i + \Delta_k & r\gamma + \xi t\delta \end{pmatrix} \qquad \text{for } k \in \{1,\ldots,t\} \text{ — columns: } \eta_{1,1}, \eta_{2,i}, \theta, \lambda_k, \mu$$

are replaced by the matrices:

$$\begin{pmatrix} 1 & \beta & \sum_{i\in I_k} a_{k,i} z_i & \gamma + \xi\delta \\ r & s & \sum_{i\in I_k} a_{k,i} r z_i + \Delta_k & r\gamma + \xi t\delta \end{pmatrix} \qquad \text{for } k \in \{1,\ldots,t\} \text{ — columns: } \eta_1, \theta, \lambda_k, \mu.$$

□

B.3 Extension to Systems of Quadratic Pairing Product Equations

In this section, we first show how to use our constructions of SPHF to deal with system of linear pairing product equations in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. Then we slightly change the language of the SPHF to be able to deal with a quadratic pairing product equations.

Lifting to $\mathbb{G}_T$ and system of linear pairing equations. The main idea for system of linear pairing product equations consists in "lifting" all commitments to $\mathbb{G}_T$: roughly, we transform any commitment of a value X in $\mathbb{G}_1$ or $\mathbb{G}_2$ into a commitment of $e(X, A)$ or $e(A, X)$, with A a constant in $\mathbb{G}_2$ or $\mathbb{G}_1$. Then a linear pairing product equation becomes a simple multi-exponentiation equation in $\mathbb{G}_T$.

Notations. Let $C_{\omega,1},\ldots,C_{\omega,n_\omega}$, for $\omega \in \{1,2\}$, be commitments of values $Y_{\omega,1},\ldots,Y_{\omega,n_\omega}$ in $\mathbb{G}_\omega$, with public key $\mathsf{ek}_\omega = (\mathbb{G}_\omega, g_{\omega,1}, g_{\omega,2}, c_\omega, d_\omega, h_\omega)$. Let $C_{T,1},\ldots,C_{T,n_T}$ be commitments of values $\mathcal{Z}_1,\ldots,\mathcal{Z}_{n_T}$ in $\mathbb{G}_T$, with public key $\mathsf{ek}_T = (\mathbb{G}_T, g_{T,1}, g_{T,2}, c_T, d_T, h_T)$. We are interested in system of linear pairing product equations which are systems of the form:

$$\begin{cases} \displaystyle\prod_{i\in I_{1,1}} e(Y_{1,i}, A_{2,1,i}) \times \prod_{i\in I_{2,1}} e(A_{1,1,i}, Y_{2,i}) \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{1,i}} = B_1 & (2.1) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ \displaystyle\prod_{i\in I_{1,t}} e(Y_{1,i}, A_{2,t,i}) \times \prod_{i\in I_{2,t}} e(A_{1,t,i}, Y_{2,i}) \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{t,i}} = B_t & (2.t) \end{cases}$$

with $I_{\omega,i} \subseteq \{1,\ldots,n_\omega\}$ (for $\omega \in \{1,2,T\}$), $A_{1,k,i} \in \mathbb{G}_1$, $A_{2,k,i} \in \mathbb{G}_2$, $b_{k,i} \in \mathbb{Z}_p$ and $B_k \in \mathbb{G}_T$. We suppose that $A_{\omega,k,i} = 1$ for $i \in I_{\omega,i}$ (for $\omega \in \{1,2\}$) and $b_{k,i} = 0$ for $i \in I_{T,k}$, and $A_{\omega,k,i} \neq 1$, $b_{k,i} \neq 0$ otherwise.

Naive lifting. The basic idea is to lift the linear pairing product equations to multi-exponentiation equations in $\mathbb{G}_T$. Let us write $C_{\omega,i} = (\mathbf{u}_{\omega,i} = (g_{\omega,1}^{r_{\omega,i}}, g_{\omega,2}^{r_{\omega,i}}), e_{\omega,i} = Y_{\omega,i} \cdot h_\omega^{r_{\omega,i}}, v_{\omega,i} = (c_\omega d_\omega^{\xi_{\omega,i}})^{r_{\omega,i}})$, with $\xi_{\omega,i} = \mathfrak{H}_K(\ell_{\omega,i}, \mathbf{u}_{\omega,i}, e_{\omega,i})$, for $\omega \in \{1,2\}$. We can lift these commitments to commitments $C'_{1,k,i}$ and $C'_{2,k,i}$ of $Y'_{1,k,i} = e(Y_{1,i}, A_{2,k,i})$ and of $Y'_{2,k,i} = e(A_{1,1,i}, Y_{2,i})$:

$$\begin{aligned}
C'_{1,k,i} &= (\mathbf{u}'_{1,k,i} = (e(u_{1,i,1}, A_{2,k,i}), e(u_{1,i,2}, A_{2,k,i})), e'_{1,k,i} = e(e_{1,i}, A_{2,k,i}), v'_{1,k,i} = e(v_{1,i}, A_{2,k,i})) \\
&= (\mathbf{u}'_{1,k,i} = (g'^{r_{1,i}}_{1,k,i,2}, g'^{r_{2,i}}_{1,k,i,1}), e'_{1,k,i} = Y'_{1,k,i} \cdot h'^{r_{1,i}}_{1,k,i}, v'_{1,k,i} = (c'_{1,k,i} d'^{\xi_{1,i}}_{1,k,i})^{r_{1,i}}) \\
C'_{2,k,i} &= (\mathbf{u}'_{2,k,i} = (e(A_{1,k,i}, u_{2,i,1}), e(A_{1,k,i}, u_{2,i,2})), e'_{2,k,i} = e(A_{1,k,i}, e_{2,i}), v'_{2,k,i} = e(A_{2,k,i}, v_{2,i})) \\
&= (\mathbf{u}'_{2,k,i} = (g'^{r_{1,i}}_{2,k,i,2}, g'^{r_{2,i}}_{2,k,i,1}), e'_{2,k,i} = Y'_{2,k,i} \cdot h'^{r_{2,i}}_{2,k,i}, v'_{2,k,i} = (c'_{2,k,i} d'^{\xi_{2,i}}_{2,k,i})^{r_{2,i}})
\end{aligned}$$

with public keys

$$\begin{aligned}
\mathsf{ek}'_{1,k,i} =& (\mathbb{G}_1, g'_{1,k,i,1} = e(g_{1,1}, A_{2,k,i}), g'_{1,k,i,2} = e(g_{1,2}, A_{2,k,i}), \\
& c'_{1,k,i} = e(c_1, A_{2,k,i}), d'_{1,k,i} = e(d_1, A_{2,k,i}), h'_{1,k,i} = e(h_1, A_{2,k,i})) \\
\mathsf{ek}'_{2,k,i} =& (\mathbb{G}_2, g'_{2,k,i,1} = e(A_{1,k,i}, g_{2,1}), g'_{2,k,i,2} = e(A_{1,k,i}, g_{2,2}), \\
& c'_{2,k,i} = e(A_{1,k,i}, c_2)1, d'_{2,k,i} = e(A_{1,k,i}, d_2), h'_{2,k,i} = e(A_{1,k,i}, h_2)).
\end{aligned}$$

It is easy to see that the original commitments $C_{1,i}, C_{2,i}, C_{T,i}$ are valid and satisfy the original system of equations (2.1), ..., (2.t) if and only if the new commitments $C'_{1,k,i}, C'_{2,k,i}, C_{T,i}$ are valid and satisfy the following system:

$$\begin{cases}
\prod_{i\in I_{1,1}} Y'_{1,1,i} \times \prod_{i\in I_{2,1}} Y'_{2,1,i} \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{1,i}} = B_1 & (3.1) \\
\qquad\qquad\qquad\qquad\qquad\qquad \vdots & \\
\prod_{i\in I_{1,t}} Y'_{1,t,i} \times \prod_{i\in I_{2,t}} Y'_{2,t,i} \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{t,i}} = B_t. & (3.t)
\end{cases}$$

Optimizations. The previous approach may be optimized in different ways: operations in $\mathbb{G}_T$ are a lot more costly than operations in $\mathbb{G}_1$, so one should postpone the pairing computations as much as possible to do exponentiations in $\mathbb{G}_1$. It also verifies the validity of all $C'_{\omega,k,i}$, whereas it is sufficient to verify only $C'_{\omega,k,i}$ for one k for each ω, i, or simply to verify the validity of the original commitments. If one wants to reduce the communication cost one can straightforwardly adapt the previous techniques on multi-exponentiations where we reuse randomness.

Extension to quadratic equations. We would like to be able to do SPHF for system of quadratic pairing product equations of this form:

$$\begin{cases}
\prod_{i\in I_{1,1}} e(Y_{1,i}, A_{2,1,i}) \times \prod_{i\in I_{2,1}} e(A_{1,1,i}, Y_{2,i}) \times \prod_{i\in I_{3,1}} \prod_{j\in I_{4,1}} e(Y_{1,i}, Y_{2,j})^{a_{1,i}} \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{1,i}} = B_1 & (4.1) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots & \\
\prod_{i\in I_{1,t}} e(Y_{1,i}, A_{2,t,i}) \times \prod_{i\in I_{2,t}} e(A_{1,t,i}, Y_{2,i}) \times \prod_{i\in I_{3,t}} \prod_{j\in I_{4,t}} e(Y_{1,i}, Y_{2,j})^{a_{t,i}} \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{t,i}} = B_t & (4.t)
\end{cases}$$

which is a slight generalization of Groth-Sahai pairing product equations where elements of $\mathbb{G}_T$ can also be unknown.

Unfortunately, we have not found a way to do directly such an SPHF. However, if we accept to slightly change the languages, we can do quadratic equations. The idea is the following: a word in the language is no more just the list of

commitments $(C_{\omega,i})$ but it is this list together with commitments $C''_{i,j}$ of $Y''_{i,j} = Y_{2,j}^{r_{1,i}}$. And we can linearize the system of equations to:

$$e(u_{1,i}, Y_{2,j}) = e(g_{1,1}, Y''_{2,j}) \tag{6}$$

for any i, j and

$$\begin{cases} \prod_{i\in I_{1,1}} e(Y_{1,i}, A_{2,1,i}) \times \prod_{i\in I_{2,1}} e(A_{1,1,i}, Y_{2,i}) \times \prod_{i\in I_{3,1}} \prod_{j\in I_{4,1}} \left(e(e_{1,i}, Y_{2,j})^{a_{1,i}} e(h_1, Y''_{i,j})^{-a_{1,i}}\right) \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{1,i}} = B_1 & (6.1) \\ \qquad\qquad \vdots \\ \prod_{i\in I_{1,t}} e(Y_{1,i}, A_{2,t,i}) \times \prod_{i\in I_{2,t}} e(A_{1,t,i}, Y_{2,i}) \times \prod_{i\in I_{3,t}} \prod_{j\in I_{4,t}} \left(e(e_{1,i}, Y_{2,j})^{a_{t,i}} e(h_1, Y''_{i,j})^{-a_{t,i}}\right) \times \prod_{i\in I_{T,i}} \mathcal{Z}_i^{b_{t,i}} = B_t & (6.t) \end{cases}$$

The first equations (6) are used to ensure $Y''_{i,j} = Y_{2,j}^{r_{1,i}}$. And, when Equations (6.1), ..., (6.t) are verified, then Equations (4.1), ..., (4.t) are verified because:

$$\begin{aligned} e(e_{1,i}, Y_{2,j})^{a_{k,i}} e(h_1, Y''_{i,j})^{-a_{k,i}} &= e(Y_{1,i} \cdot h^{r_{1,i}}, Y_{2,j})^{a_{k,i}} e(h_1, Y_{2,j}^{r_{1,i}})^{-a_{k,i}} \\ &= e(Y_{1,i}, Y_{2,j})^{a_{1,i}} e(h, Y_{2,j})^{r_{1,i}a_{k,i}} e(h, Y_{2,j})^{-r_{1,i}a_{k,i}} = e(Y_{1,i}, Y_{2,j}). \end{aligned}$$

C Proof of Theorem 9

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.

C.1 Game 1

We first modify the way Execute-queries are answered: we replace C and C' by encryptions of a fixed message $M_0 = (\mathsf{aux}, W)$ such that W is not in the induced language. Since the hashing keys are known, the common session key is computed as

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

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 $\mathsf{IND} - \mathsf{CPA}$ property of the encryption scheme, for each Execute-query. Using a classical hybrid technique, one thus gets $|\mathsf{Adv}_1(\mathcal{A}) - \mathsf{Adv}_0(\mathcal{A})| \leq \mathsf{negl}()$.

C.2 Game 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: $|\mathsf{Adv}_2(\mathcal{A}) - \mathsf{Adv}_1(\mathcal{A})| \leq \mathsf{negl}()$.

C.3 Game 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 $\mathsf{aux} = (\mathsf{priv}, priv')$ and W used by the adversary. Then
  - if they are consistent with the receiver's values —event Ev— one declares that $\mathcal{A}$ succeeds (saying that $b' = b$) and terminates the game;
  - if they are not 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 Game 5). 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: $\mathsf{Adv}_2(\mathcal{A}) \leq \mathsf{Adv}_3(\mathcal{A}) + \mathsf{negl}()$.

C.4 Game 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 $\Pi^t_{U'}$ partnered with Π^s_U that receives this flow, then set the key identical to the key for $\Pi^t_{U'}$;
- 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: $|\mathsf{Adv}_4(\mathcal{A}) - \mathsf{Adv}_3(\mathcal{A})| \leq \mathsf{negl}()$.

C.5 Game 5

We now modify the way one answers the Send_0-queries: instead of committing the correct values, one does as in Game 1 for $\mathsf{Execute}$-queries, and commits M_0. Since for simulating the Send_1-queries decryptions are required, indistinguishability relies on the $\mathsf{IND-CCA}$ security of the encryption scheme: $|\mathsf{Adv}_5(\mathcal{A}) - \mathsf{Adv}_4(\mathcal{A})| \leq \mathsf{negl}()$.

C.6 Game 6

For all the hashing and projection keys, we now use the dummy private inputs. Since we consider we have a type II $\mathsf{KV-SPHF}$, the distributions of these keys are (at least computationally) independent of the auxiliary 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 committed $\mathsf{aux} = (\mathsf{pub}, \mathsf{priv}, priv')$ and W that are consistent with the receiver's values. 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:

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

where k lists all the Send_1-queries with adversary-generated messages 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 $\mathrm{L}_{\mathsf{aux}}$. Then, by combining all the inequalities, one gets

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

D Efficient Instantiation of the One-Round UC PAKE of Katz-Vaikuntanathan

In this section, we show a fairly efficient instantiation of the one-round UC PAKE proposed by Katz-Vaikuntanathan in [KV11] (denoted KV-UC-PAKE in the sequel). This instantiation is several hundreds times more efficient than a naive instantiation of their scheme. Since this is not the main purpose of our article, we do not recall precisely the UC model nor the definitions of the primitives we use, and we report the reader to [KV11].

D.1 Recall of the Scheme

The KV-UC-PAKE is an extension of the one-round PAKE described in Section 2.5. In addition the projection key hp and the commitment C, each user also sends a simulation-sound extractable non-interactive zero-knowledge proof (SSENIZK) that he knows the secret key hk. Actually, in [KV11], Katz-Vaikuntanathan commit the hashing key hk and do a simulation-sound non-interactive zero-knowledge proof (SSNIZK) that they really have committed a valid hashing key hk corresponding to hp.But, since the SSNIZK they use (and we use) are also extractable, the commitment of hk is actually useless.

As in [KV11], we suppose we have a commitment scheme $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Decommit})$ and an associated type III KV-SPHF for the language:

$$\mathrm{L}_{\mathsf{pw}} = \{(\ell, C) \mid \exists r, C = \mathsf{Commit}(\ell, \mathsf{pw}; r)\}$$

Let us consider a SSENIZK for the following NP language:

$$\mathrm{L}^* = \{\mathsf{hp} \mid \exists \mathsf{hk}, \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \perp, W)\}.$$

Let σ be a common reference string for this SSENIZK, and Prove, Ver and Ext be three polynomial algorithms such that: $\mathsf{Prove}(\sigma, \mathsf{hp}, \mathsf{hk})$ returns a proof π that hp is in L, $\mathsf{Ver}(\sigma, \mathsf{hp}, \pi)$ returns 1 if and only if the proof π is correct and $\mathsf{Ext}(\sigma, \tau, \mathsf{hp}, \pi)$ extracts a hashing key hk corresponding to hp from the proof π using a trapdoor τ (in σ).

D.2 Naive Instantiation

In [KV11], Katz-Vaikuntanathan suggest to use a double ElGamal encryption plus an SSNIZK for the commitment scheme and to use Groth-Sahai methodology [GS08] to construct an SSENIZK for L^*. We can obviously improve this instantiation by using the commitment scheme and its associated SPHF we proposed in Section 2.5. But in both cases, the SSENIZK need to be done by considering hk as a bit-string and committing each bit of hk separately, because hk is a tuple of scalars and not a tuple of group elements.

Therefore, if hk contains k scalars ($k = 4$ with commitments and SPHF of Katz-Vaikuntanathan and $k = 6$ with our commitments and SPHF), with Groth-Sahai methodology, we need at least $k \times 512$-group elements just for the commitments of the bits of hk (with 128-bit security and so 256-bit scalars) !

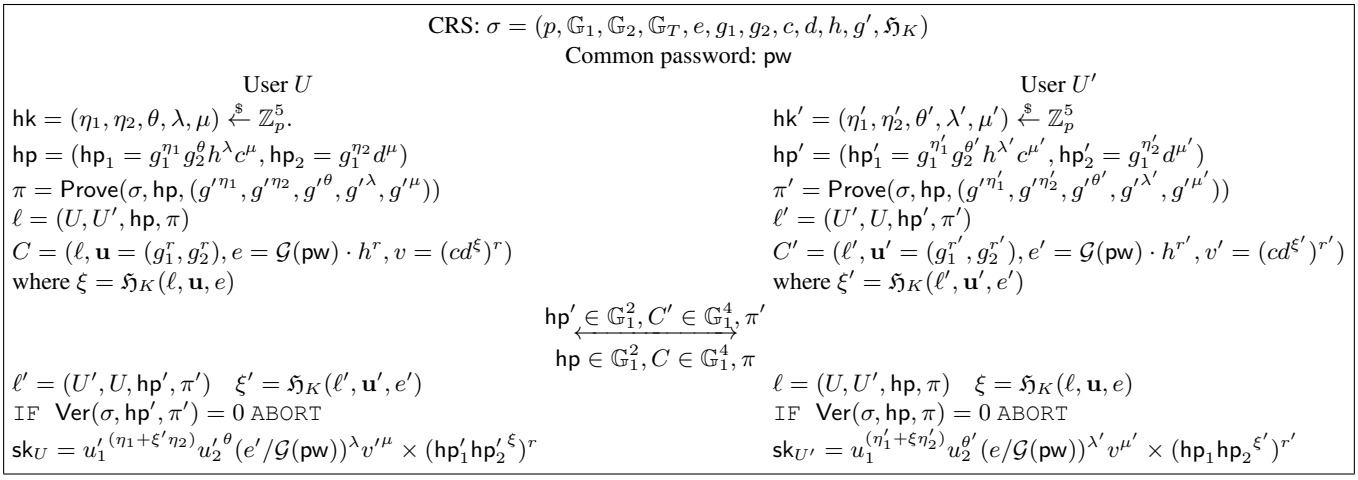

Fig. 3. KV-UC-PAKE based on SXDH with efficient SSENIZK

D.3 Efficient Instantiation

In this section, we show that, if we slightly tweak the SPHF, we can do a SSENIZK on group elements, which significantly reduces its cost.

Let us consider a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$. We do all commitments in $\mathbb{G}_1$, using a CS commitment as before. But we use the following SPHF: the hashing key still consists of a random tuple $\mathsf{hk} = (\eta_1, \eta_2, \theta, \lambda, \mu) \xleftarrow{\$} \mathbb{Z}_p^5$ and the associated projection key is the same as before: $\mathsf{hp} = (\mathsf{hp}_1 = g_1^{\eta_1} g_2^{\theta} h^{\lambda} c^{\mu}, \mathsf{hp}_2 = g_1^{\eta_2} d^{\mu}) \in \mathbb{G}_1^p$. But the hash value is now:

$$H \stackrel{\text{def}}{=} \mathsf{Hash}(\mathsf{hk}, \mathrm{L}_m, C) \stackrel{\text{def}}{=} e\left(u_1^{(\eta_1 + \xi\eta_2)} u_2^{\theta} (e/M)^{\lambda} v^{\mu}, g_2\right) = e\left((\mathsf{hp}_1 \mathsf{hp}_2^{\xi})^r, g_2\right) \stackrel{\text{def}}{=} \mathsf{ProjHash}(\mathsf{hp}, \mathrm{L}_m, C, r) \stackrel{\text{def}}{=} H',$$

with g' a generator of $\mathbb{G}_2$. It is clear that this new SPHF is also a perfectly smooth type III $\mathsf{KV}-\mathsf{SPHF}$.

The idea is that we can compute H in a third way:

$$H = e(u_1, H_1)\, e(u_1, H_2)^{\xi}\, e(u_2, \Theta)\, e(e/M, \Lambda)\, e(v, \Gamma)$$

with $H_1 = g'^{\eta_1}$, $H_2 = g'^{\eta_2}$, $\Theta = g'^{\theta}$, $\Lambda = g'^{\lambda}$, $\Gamma = g'^{\mu}$. Contrary to the formula using hp (corresponding to $\mathsf{ProjHash}$), we can compute H for any C (even if C is invalid or corresponds to a commitment of a value different from M); and contrary to the formula using hk (corresponding Hash), we need to know five group elements $(H_1, H_2, \Theta, \Lambda, \Gamma)$ instead of five scalars. Therefore, using this trick, we can use a SSENIZK for the language L^* defined as the set of $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2)$ such that:

$$\exists (H_1, H_2, \Theta, \Lambda, \Gamma) \in \mathbb{G}_2^5, e(\mathsf{hp}_1, g') = e(g_1, H_1)\, e(g_2, \Theta)\, e(h, \Lambda)\, e(c, \Gamma) \text{ and } e(\mathsf{hp}_2, g') = e(g_1, H_2)\, e(d, \Gamma).$$

It is easy to see that if $(H_1, H_2, \Theta, \Lambda, \Gamma)$ is a valid witness, there exists $\mathsf{hk} = (\eta_1, \eta_2, \theta, \lambda, \mu) \in \mathbb{Z}^5$ such that $H_1 = g'^{\eta_1}$, $H_2 = g'^{\eta_2}$, $\Theta = g'^{\theta}$, $\Lambda = g'^{\lambda}$, $\Gamma = g'^{\mu}$. Since witnesses is a tuple of 5 elements, the associated SSENIZK is a lot more efficient, and the whole construction would require 8 group elements in $\mathbb{G}_1$, 14 in $\mathbb{G}_2$ and 4 in $\mathbb{Z}_p$. A similar instantiation in a symmetric group would require around 39 group elements to be compared with the 60 required by [JR12]. In both cases we are roughly 33% more efficient than the best current instantiation. The protocol is depicted in Figure 3.

Smooth Projective Hash Cheat Sheet

Simplistic Vision of SPHF

Given a word $W \in X$ a projection key hp generated honestly from hk, one should have:

If one possesses a witness w that $W \in L \subset X$, $\mathsf{ProjHash}(\mathsf{hp}, W; w) = \mathsf{Hash}(\mathsf{hk}, W)$,
If L is a hard-subset from X, then without w, H is computationally-indistinguishable from a random value,
If $W \notin L$, H is indistinguishable from a random value.

Different kinds of SPHF

	hp sent before W	hp depends on W
$\mathsf{KV-SPHF}$	✓	$\perp$
$\mathsf{CS-SPHF}$	$\perp$	$\perp$
$\mathsf{GL-SPHF}$	$\perp$	✓

There is a trade-off between when hp has to be generated and its size.
A KV-SPHF is a CS-SPHF, and a CS-SPHF is a GL-SPHF.

Fig. 4. The three historic kinds of SPHF

	Projection Key
Type I	Language-Dependent
Type II	Language-Indistinguishable
Type III	Language-Independent

One can expect smaller type to allow more efficient instantiation.
Of course a Type III SPHF is also Type II, and a type II is also Type I.

Fig. 5. A different classification based on the link between the Projection Key and the Language

Manageable Languages

One can always proceed to conjunctions of languages. Disjunctions can also be done but they may require some extra work especially in the KV/CS setting.

Without Pairings:
- El-Gamal / Linear Encryption
- (Linear) Cramer-Shoup
- Linear Multi-Exponentiation equation (where the variables are the group elements)

With Pairings:
- Linear Pairing Equation (possibly with variables in $\mathbb{G}_T$)
- Quadratic Pairing Equation (possibly with variables in $\mathbb{G}_T$)

Interactive Protocols

- PAKE: $\mathsf{pub} = \emptyset$, $\mathsf{priv} = priv = W = W'$,
- Secret Handshake: $\mathsf{pub} = (\mathsf{id}, \mathsf{id}')$, $\mathsf{priv} = priv = \mathsf{vk}$,
- Anonymous Secret Handshake: $\mathsf{priv} = (\mathsf{vk}, \mathsf{id})$, $priv = (\mathsf{vk}, \mathsf{id}')$,
- LAKE: $\mathsf{pub}, \mathsf{priv}, priv$.