Distributed Smooth Projective Hashing and its Application to Two-Server PAKE[*]

Franziskus Kiefer and Mark Manulis

Department of Computing, University of Surrey, UK
`mail@franziskuskiefer.de`, `mark@manulis.eu`

Abstract. Smooth projective hash functions have been used as building block for various cryptographic applications, in particular for password-based authentication.

In this work we propose the extended concept of *distributed* smooth projective hash functions where the computation of the hash value is distributed across n parties and show how to instantiate the underlying approach for languages consisting of Cramer-Shoup ciphertexts.

As an application of distributed smooth projective hashing we build a new framework for the design of two-server password authenticated key exchange protocols, which we believe can help to "explain" the design of earlier two-server password authenticated key exchange protocols.

Keywords: Smooth Projective Hash Functions, Two-Server PAKE

1 Introduction

Smooth projective hashing allows to compute the hash value of an element from a set in two different ways: either by using a secret hashing key on the element, or utilising the public projection key and some secret information proving that the particular element is part of a specific subset under consideration. In addition, smooth projective hash values guarantee to be uniformly distributed in their domain as long as the input element is not from a specific subset of the input set. These features make them a quite popular building block in many protocols such as CCA-secure public key encryption, blind signatures, password authenticated key exchange, oblivious transfer, zero-knowledge proofs, commitments and verifiable encryption.

Smooth projective hash functions (SPHF) are due to Cramer and Shoup [10] who used them to construct CCA-secure public key encryption schemes and analyse mechanisms from [9]. The first use of SPHFs in the construction of a password authenticated key exchange (PAKE) protocol is due to Gennaro and Lindell [11], who introduced additional requirements to the SPHF such as pseudorandomness that was later extended in [15]. The SPHF-based approach taken in [11] was further helpful in the "explanation" of the KOY protocol from [14], where those functions were implicitly applied.

Abdalla et al. [1] introduced conjunction and disjunction of languages for smooth projective hashing that were later used in the construction of blind signatures [7,5], oblivious signature-based envelopes [7], and authenticated key exchange protocols for algebraic languages [4]. Blazy et al. [7] demonstrate more general use of smooth projective hashing in designing round-optimal privacy-preserving interactive protocols.

We extend this line of work by considering divergent parametrised languages in one smooth projective hash function that allows multiple parties to jointly evaluate the result of the function. We propose the notion of (distributed) extended smooth projective hashing that enables joint hash computation for special languages. Further, we propose a new two-server password authenticated key exchange framework using the new notion of distributed smooth projective hashing and show how it helps to explain the protocol from [13]. Actually, the authors of [2] already built a group PAKE protocol using smooth projective hashing in a multi-party party protocol. However, they assume a ring structure such that the smooth projective hashing is only used between two parties.

[*]A shortened version of this paper with the name "Distributed Smooth Projective Hashing and its Application to Two-Server Password Authenticated Key Exchange" appears in the proceedings of the 12th International Conference on Applied Cryptography and Network Security (ACNS 2014), June 10–13, 2014, Lausanne, Switzerland. This is the full version.

Organisation We start by recalling smooth projective hash functions and introduce useful definitions in Section 2. Our first contribution is the definition of an extended smooth projective hash function SPHF^x that handles divergent parametrised languages in Section 3. Then we show how to distribute their computation between multiple parties, introducing distributed SPHF^x in Section 3.1 and give a concrete instantiation in Section 3.3. Finally, we propose a two-server PAKE framework in Section 4 and analyse the two-server KOY protocol using a variant of distributed SPHF^x in Section 4.2.

2 Smooth Projective Hash Functions

First, we recall definitions from [5] for classical SPHF with some minor changes. We stick with the framework from [5, Section 3] on cyclic groups $\mathbb{G}$ of prime order and focus on languages of ciphertexts. This seems reasonable since it is the preferred setting and allows a comprehensible description. An extension to graded rings and general languages should be possible and is left open for future work.

A language $L_{\mathtt{aux}}$ is indexed by a parameter $\mathtt{aux}$, consisting of global public information and secret variable information $\mathtt{aux}'$. In our setting of languages of ciphertexts the public part of $\mathtt{aux}$ is essentially a common reference string $\mathtt{crs}$ containing the public key $\mathtt{pk}$ of the used encryption scheme. The secret part $\mathtt{aux}'$ contains the message that should be encrypted. By π we denote the $\mathtt{crs}$ trapdoor, the secret key to $\mathtt{pk}$. We denote $\mathcal{L}$ the encryption scheme used to generate words. Unless stated otherwise we assume that $\mathcal{L}$ is a labelled CCA-secure encryption scheme.

Definition 1 (Languages of Ciphertexts). *Let $L_{\mathtt{aux}} \subseteq \mathcal{S}\mathrm{et}$ denote the language of ciphertext under consideration. A ciphertext C is in the language $L_{\mathtt{aux}}$ if $C \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\ell, \mathtt{aux}'; w)$ for $\mathtt{aux} = (\mathtt{pk}, \mathtt{aux}')$. Formally, a word C is in the language $L_{\mathtt{aux}}$ if and only if $\exists \lambda \in \mathbb{Z}_p^{1\times k}$ such that $\Theta_{\mathtt{aux}}(C) = \lambda \odot \Gamma(C)$, where $\Gamma : \mathcal{S}\mathrm{et} \mapsto \mathbb{G}^{k\times n}$ and $\Theta_{\mathtt{aux}} : \mathcal{S}\mathrm{et} \mapsto \mathbb{G}^{1\times n}$ for integers k, n.*

We use the notation $\odot$ and common matrix and vector operations on it from [5]: for $a \in \mathbb{G},\ r \in \mathbb{Z}_p :\ a \odot r = r \odot a = a^r \in \mathbb{G}$ (cf. Appendix C).

Definition 2 (SPHF [5]). *Let $L_{\mathtt{aux}}$ denote a language such that $C \in L_{\mathtt{aux}}$ if there exists a witness w proving so. A* smooth projective hash function *for ciphertext language $L_{\mathtt{aux}}$ consists of the following four algorithms:*

- $\mathtt{KGen_H}(L_{\mathtt{aux}})$ *generates a hashing key $\mathtt{k_h} \in_R \mathbb{Z}_p^{1\times n}$ for language $L_{\mathtt{aux}}$.*
- $\mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}, C)$ *derives the projection key $\mathtt{k_p} = \Gamma(C) \odot \mathtt{k_h} \in \mathbb{G}^{k\times 1}$, possibly depending on C.*
- $\mathtt{Hash}(\mathtt{k_h}, L_{\mathtt{aux}}, C)$ *outputs the hash value $h = \Theta_{\mathtt{aux}}(C) \odot \mathtt{k_h} \in \mathbb{G}$.*
- $\mathtt{PHash}(\mathtt{k_p}, L_{\mathtt{aux}}, C, w)$ *returns the hash value $h = \lambda \odot \mathtt{k_p} \in \mathbb{G}$, with $\lambda = \Omega(w, C)$ for some $\Omega : \{0,1\}^* \mapsto \mathbb{G}^{1\times k}$.*

A SPHF has to fulfil the following three properties (formal definitions follow):

- *Correctness*: If $C \in L$, with w proving so, then $\mathtt{Hash}(\mathtt{k_h}, L_{\mathtt{aux}}, C) = \mathtt{PHash}(\mathtt{k_p}, L_{\mathtt{aux}}, C, w)$.
- *Smoothness*: If $C \notin L_{\mathtt{aux}}$, the hash value h is statistically indistinguishable from a random element in $\mathbb{G}$.
- *Pseudorandomness*: If $C \in L_{\mathtt{aux}}$, the hash value h is indistinguishable from a random element in $\mathbb{G}$.[1]

In a nutshell, smoothness ensures that the hash value always looks random in $\mathbb{G}$ when computed on an element not in the language, while pseudorandomness ensures that it looks random in $\mathbb{G}$ when computed on an element in the language. The authors of [6] identify three different SPHF classes: word-independent key and adaptive smoothness (KV-SPHF, first proposed in [15]), word-independent key and non-adaptive smoothness (CS-SPHF, first proposed in [10]), and word-dependent key (GL-SPHF, first proposed in [11]).

In this work we focus on the strongest notion behind KV-SPHF: *word-independent key* with *adaptive smoothness*. Unless stated otherwise all SPHFs in the following are KV-SPHFs where the projection key is

[1]Note that this is not always a requirement or even possible. But as languages of labelled CCA-secure ciphertexts are hard-on-average problems the corresponding SPHF is also pseudorandom.

independent of the ciphertext. This property enables our construction of extended SPHFs. The corresponding notion of adaptive smoothness with word-independent keys is defined as follows. For any function $f : \mathcal{S}\mathrm{et} \setminus L_{\mathtt{aux}} \mapsto \mathbb{G}^{l\times 1}$ the following distributions are statistically ε-close:

$$\{(\mathtt{k_p}, h) \mid \mathtt{k_h} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathtt{aux}}); \mathtt{k_p} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}); h \leftarrow \mathtt{Hash}(\mathtt{k_h}, L_{\mathtt{aux}}, f(\mathtt{k_p}))\}$$
$$\overset{\varepsilon}{\equiv} \{(\mathtt{k_p}, h) \mid \mathtt{k_h} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathtt{aux}}); \mathtt{k_p} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}); h \in_R \mathbb{G}\}$$

Gennaro and Lindell [11] introduced pseudorandomness of SPHFs to show that `Hash` and `PHash` are the only way to compute the hash value even though the adversary knows some tuples $(\mathtt{k_p}, C, \mathtt{Hash}(\mathtt{k_h}, L_{\mathtt{aux}}, C))$ for $C \in L_{\mathtt{aux}}$. A SPHF is pseudorandom if the hash values produced by `Hash` and `PHash` are indistinguishable from random without the knowledge of the uniformly chosen hash key $\mathtt{k_h}$ or a witness w, i.e. for all $C \in L_{\mathtt{aux}}$ the following distributions are computationally ε-close:

$$\{(\mathtt{k_p}, C, h) \mid \mathtt{k_h} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathtt{aux}}); \mathtt{k_p} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}); h \leftarrow \mathtt{Hash}(\mathtt{k_h}, L_{\mathtt{aux}}, C)\}$$
$$\overset{\varepsilon}{\equiv} \{(\mathtt{k_p}, C, h) \mid \mathtt{k_h} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathtt{aux}}); \mathtt{k_p} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}); h \in_R \mathbb{G}\}$$

To define pseudorandomness of a SPHF we use an experiment based on those from [11, Corollary 3.3] and [15].

Definition 3 (SPHF Pseudorandomness). *For all PPT algorithms $\mathcal{A}$ there exists a negligible function $\varepsilon(\cdot)$ such that*

$$\left|\Pr[\mathcal{A}^{\mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\cdot),\mathtt{Hash}(\cdot)} = 1] - \Pr[\mathcal{A}^{\mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\cdot),\mathcal{U}(\cdot)} = 1]\right| < \varepsilon(\lambda).$$

– $\mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\ell, \mathtt{aux})$ *with* $\mathtt{aux} = (\mathtt{pk}, \mathtt{aux}')$ *returns elements* $C \in L_{\mathtt{aux}}$ *encrypting* $\mathtt{aux}'$ *using* $\mathtt{pk}$*, label* ℓ *and encryption algorithm* $\mathcal{L}$*.*
– $\mathtt{Hash}(C)$ *returns* $(\mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}, C), \mathtt{Hash}(\mathtt{k_h}, L_{\mathtt{aux}}, C))$ *for fresh* $\mathtt{k_h} \leftarrow \mathtt{KGen_H}\ (L_{\mathtt{aux}})$ *if* C *has been output by* $\mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}$*, nothing otherwise.*
– $\mathcal{U}(C)$ *returns* $(\mathtt{KGen_P}(\mathtt{k_h}, L_{\mathtt{aux}}, C), h)$ *for fresh* $\mathtt{k_h} \leftarrow \mathtt{KGen_H}(L_{\mathtt{aux}})$ *and random* $h \in \mathbb{G}$ *if* C *has been output by* $\mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}$*, nothing otherwise.*

While the authors of [5,6] have skipped the proof of pseudorandomness as it is straightforward, we want to briefly give an intuition why their SPHF framework is pseudorandom. The reasoning for pseudorandomness of SPHFs is actually easy and always follows the same approach given in [11]. By replacing the correct ciphertexts in the simulation with ciphertexts $C \notin L_{\mathtt{aux}}$ we can use the smoothness of SPHFs to show their indistinguishability. The replacement itself is covered by the hard-on-average subset membership problem, in the case of ciphertexts their CCA-security. In [15] pseudorandomness in the case of hash key and ciphertext reuse is added. We discuss this extension when defining *concurrent* pseudorandomness of our extended smooth projective hash functions in the next section.

Encryption Schemes & SPHFs We use SPHFs on labelled *Cramer-Shoup (CS)* encryptions throughout this work as an example, i.e. $\mathcal{L} = \mathrm{CS}$. Thus, we briefly recall its definition. Let $C = (\ell, \boldsymbol{u}, e, v) \leftarrow \mathtt{Enc}^{\mathrm{CS}}_{\mathtt{pk}}(\ell, m; r)$ with $\boldsymbol{u} = (u_1, u_2) = (g_1^r, g_2^r)$, $e = h^r g_1^m$ and $v = (cd^\xi)^r$ with $\xi = H_k(\ell, \boldsymbol{u}, e)$ denote a labelled Cramer-Shoup ciphertext. We assume $m \in \mathbb{Z}_p$ and $\mathbb{G}$ is a cyclic group of prime order p with generators g_1 and g_2 such that $g_1^m \in \mathbb{G}$. The CS public key is defined as $\mathtt{pk} = (p, \mathbb{G}, g_1, g_2, c, d, H_k)$ with $c = g_1^{x_1} g_2^{x_2}, d = g_1^{y_1} g_2^{y_2}, h = g_1^z$ and hash function H_k such that $\mathtt{dk} = (x_1, x_2, y_1, y_2, z)$ denotes the decryption key. Decryption is defined as $g_1^m = \mathtt{Dec}^{\mathrm{CS}}_{\mathtt{dk}}(C) = e/u_1^z$ if $u_1^{x_1+y_1\cdot\xi'} u_2^{x_2+y_2\cdot\xi'} = v$ with $\xi' = H_k(\ell, \boldsymbol{u}, e)$. Benhamouda et al. propose a new perfectly smooth SPHF for labelled Cramer-Shoup encryptions in [5]. Note that the witness for $C \in L_{\mathtt{aux}}$ is the used randomness $w = r$. The SPHF is den given by Definition 2 and the following variables: $\Gamma(C) = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \in \mathbb{G}^{2\times 5}$, $\lambda = (r, r\xi) \in \mathbb{Z}_p^{1\times 2}$ for $\Omega(r, C) = (r, r\xi)$, $\Theta_{\mathtt{aux}}(C) = (u_1, u_1^\xi, u_2, e/m, v) \in \mathbb{G}^{1\times 5}$ and $\mathtt{k_h} = (\eta_1, \eta_2, \theta, \mu, \nu) \in_R \mathbb{Z}_p^{1\times 5}$.

We further use *El-Gamal (EG)* encryptions. Let $C = (u, e) \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}}(m; r)$ with $u = g^r$ and $e = h^r g^m$ denote an El-Gamal ciphertext. Note that we assume $m \in \mathbb{Z}_p$ and $\mathbb{G}$ is a cyclic group of prime order p with generator g such that $g^m \in \mathbb{G}$. The El-Gamal public key is defined as $\mathtt{pk} = (p, \mathbb{G}, g, h)$ with $h = g^z$ such that $\mathtt{dk} = z$ denotes the decryption key. Decryption is given by $g^m = \mathtt{Dec}^{\mathrm{EG}}_{\mathtt{dk}}(C) = e/u^z$. A SPHF on El-Gamal ciphertexts can be build from Definition 2 using the following variables: $\mathit{\Gamma}(C) = (g, h)^T \in \mathbb{G}^{2\times 1}$, $\lambda = r \in \mathbb{Z}_p$ for $\mathit{\Omega}(r, C) = r$, $\mathit{\Theta}_{\mathtt{aux}}(C) = (u, e/m) \in \mathbb{G}^{1\times 2}$ and $\mathtt{k_h} = (\eta, \theta) \in_R \mathbb{Z}_p^{1\times 2}$.

3 Extended Smooth Projective Hash Functions (SPHF^x)

We introduce an extended notion of smooth projective hashing that allows us to distribute the computation of the hash value. The new notion of extended SPHF (SPHF^x) is defined in the following setting: The parameter $\mathtt{aux}$, a language is indexed with, allows us to easily describe languages that differ only in the secret part $\mathtt{aux}'$. We consider a language $L_{\mathtt{aux}}$ with words (ciphertexts) C that are ordered sets of n ciphertexts $(C_0, \dots, C_x)$. The secret variable information $\mathtt{aux}'$ is chosen from the additive group $(\mathbb{P}, +) = (\mathbb{Z}_p^+, +)$ with a function $h : \mathbb{P} \mapsto \mathbb{P}^x$. Let $L^{\mathcal{L}}_{\mathtt{aux}}$ denote the language of ciphertexts encrypting the secret part $\mathtt{aux}'$ from $\mathtt{aux}$ with the public key $\mathtt{pk}$ from $\mathtt{aux}$ using encryption scheme $\mathcal{L}$. For all $C_i, i \in \{1, \dots, x\}$ it must hold that $C_i \in L^{\mathcal{L}}_{\mathtt{aux}_i}$ where $\mathtt{aux}_i = (\mathtt{pk}, \mathtt{aux}'_i)$ with $\mathtt{aux}'_i = h(\mathtt{aux}')[i]$. For C_0 it must hold that $C_0 \in L^{\mathcal{L}}_{\mathtt{aux}}$. Furthermore, the ciphertexts must offer certain homomorphic properties such that there exists a modified decryption algorithm $\mathtt{Dec}'$ and a combining function g such that $\mathtt{Dec}'_\pi(C_0) = \mathtt{Dec}'_\pi(g(C_1, \dots, C_x))$, where π denotes the secret key for the corresponding public key $\mathtt{pk}$ from $\mathtt{crs}$.

The idea of SPHF^x is to be able to use the SPHF functionality not only on a single ciphertext, but on a set of ciphertexts with specific properties. Due to the nature of the words considered in SPHF^x they produce two different hash values. One can think of the two hash values as h_0 for C_0 and h_x for $C_1, \dots, C_x$. The hash value h_0 can be either computed with knowledge of the hash key $\mathtt{k_{h0}}$ or with the witnesses $w_1, \dots, w_x$ that $C_1, \dots, C_x$ are in $L^{\mathcal{L}}_{\mathtt{aux}_i}$ each. The hash value h_x can be computed with knowledge of the hash keys $\mathtt{k_{h1}}, \dots, \mathtt{k_{h}}_x$ or with the witness w_0 that C_0 is in $L^{\mathcal{L}}_{\mathtt{aux}}$.

Definition 4 (SPHF^x). *Let $L_{\mathtt{aux}}$ denote a language such that $C = (C_0, C_1, \dots, C_x) \in L_{\mathtt{aux}}$ if there exists a witness $w = (w_0, w_1, \dots, w_x)$ proving so and there exist functions $h(\mathtt{aux}') = (\mathtt{aux}'_1, \dots, \mathtt{aux}'_x)$ and $g : \mathbb{G}^l \mapsto \mathbb{G}^{l'}$ as described above. An extended smooth projective hash function for language $L_{\mathtt{aux}}$ with $\mathit{\Gamma} \in \mathbb{G}^{k\times n}$ consists of the following six algorithms:*

- $\mathtt{KGen_H}(L_{\mathtt{aux}})$ *generates a hashing key* $\mathtt{k_h}_i \in \mathbb{Z}_p^{1\times n}$ *for* $i \in \{0, \dots, x\}$ *and language* $L_{\mathtt{aux}}$.
- $\mathtt{KGen_P}(\mathtt{k_h}_i, L_{\mathtt{aux}})$ *derives the projection key* $\mathtt{k_p}_i = \mathit{\Gamma} \odot \mathtt{k_h}_i \in \mathbb{G}^{1\times k}$ *for* $i \in \{0, \dots, x\}$.
- $\mathtt{Hash}_x(\mathtt{k_{h0}}, L_{\mathtt{aux}}, C_1, \dots, C_x)$ *outputs hash value* $h_x = \mathit{\Theta}^x_{\mathtt{aux}}(C_1, \dots, C_x) \odot \mathtt{k_{h0}}$.
- $\mathtt{PHash}_x(\mathtt{k_{p}}_0, L_{\mathtt{aux}}, C_1, \dots, C_x, w_1, \dots, w_x)$ *returns hash value* $h_x = \prod_{i=1}^{x}(\lambda^i \odot \mathtt{k_p}_0)$, *where* $\lambda^i = \mathit{\Omega}(w_i, C_i)$.
- $\mathtt{Hash}_0(\mathtt{k_{h1}}, \dots, \mathtt{k_h}_x, L_{\mathtt{aux}}, C_0)$ *outputs hash value* $h_0 = \prod_{i=1}^{x}(\mathit{\Theta}^0_{\mathtt{aux}}(C_0) \odot \mathtt{k_h}_i) = \mathit{\Theta}^0_{\mathtt{aux}}(C_0) \odot \sum_{i=1}^{x} \mathtt{k_h}_i$.
- $\mathtt{PHash}_0(\mathtt{k_p}_1, \dots, \mathtt{k_p}_x, L_{\mathtt{aux}}, C_0, w_0)$ *returns hash value* $h_0 = \prod_{i=1}^{x}(\lambda^0 \odot \mathtt{k_p}_i)$, *with* $\lambda^0 = \mathit{\Omega}(w_0, C_0)$.

The correctness of the scheme can be easily verified by checking that $\mathtt{Hash}_x = \mathtt{PHash}_x$ and $\mathtt{Hash}_0 = \mathtt{PHash}_0$.

Security of SPHF^x We refine definitions of smoothness and pseudorandomness to account for the two different hash functions. Therefore, we add both hash values to the indistinguishable sets, as well as the vector of projection keys. We start with the smoothness of the described SPHF^x. The smoothness proven in Theorem 1 follows directly from the proof given in [5, Appendix D.3] and follows the same approach for smoothness proofs as in previous works on SPHF [5,11,15]. Recall that we are only concerned with *adaptive smoothness*. Let $\overline{\mathtt{k_p}}$ denote the vector of projection keys $\mathtt{k_p}_i$ for $i = 0, \dots, x$. For any functions f, f' to

$\mathcal{S}$et $\setminus L_{\mathsf{aux}}$ the following distributions are statistically ε-close:

$$\{(\overline{\mathtt{k_p}}, h_0, h_x) \mid h_0 \leftarrow \mathtt{Hash}_0(\mathtt{k_{h1}}, \ldots, \mathtt{k_{h}}_x, L_{\mathsf{aux}}, f(\mathtt{k_{p}}_0));\ h_x \leftarrow \mathtt{Hash}_x(\mathtt{k_{h0}}, L_{\mathsf{aux}}, f'(\mathtt{k_{p}}_1, \ldots, \mathtt{k_{p}}_x)); \forall i \in \{0, \ldots, x\} :\ \mathtt{k_{h}}_i \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathsf{aux}}); \mathtt{k_{p}}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_{h}}_i, L_{\mathsf{aux}})\}$$
$$\overset{\varepsilon}{=}\{(\overline{\mathtt{k_p}}, h_0, h_x) \mid h_0 \in_R \mathbb{G};\ h_x \in_R \mathbb{G}; \forall i \in \{0, \ldots, x\} :\ \mathtt{k_{h}}_i \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathsf{aux}}); \mathtt{k_{p}}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_{h}}_i, L_{\mathsf{aux}})\}.$$

Theorem 1 (SPHF^x Smoothness). *The SPHF^x construction from Definition 4 on cyclic groups is statistically smooth.*

Proof. We show that the logarithm of the projection keys $\overline{\mathtt{k_p}}$ and the logarithm of the hash values h_0 and h_x are defined by linearly independent equations and thus h_0 and h_x are uniform in $\mathbb{G}$, given $\overline{\mathtt{k_p}}$. In addition to this general proof we give an extended proof of the SPHF^x smoothness instantiated with labelled Cramer-Shoup encryption for better understanding in Appendix B.1. To show that $(\overline{\mathtt{k_p}}, h_0, h_x)$ is uniformly distributed in $\mathbb{G}^{k+2}$ for $C \notin L_{\mathsf{aux}}$, i.e. ε-close to $(\overline{\mathtt{k_p}}, g_0, g_x)$ for random $g_0, g_x \in_R \mathbb{G}$, we consider a word $C = (C_0, C_1, \ldots, C_x) \notin L_{\mathsf{aux}}$ and a projection key $\mathtt{k_{p}}_j = \Gamma \odot \mathtt{k_{h}}_j$ such that one C_j does not fulfill the property $C_j \in L_{\mathsf{aux}_j}$, i.e. $\exists j \in \{0, \ldots, x\}, \forall \lambda^j \in \mathbb{Z}_p^{1\times k} :\ \Theta_{\mathsf{aux}_j}(C_j) \neq \lambda^j \odot \Gamma$. From [5, Appendix D.3] it follows directly that $\Theta_{\mathsf{aux}_j}(C_j) \odot \mathtt{k_{h}}_j$ is a uniformly distributed element in $\mathbb{G}$, and thus $\Theta^x_{\mathsf{aux}}(C_1, \ldots, C_x) \odot \mathtt{k_{h0}}$ and $\prod_{i=1}^{x}(\Theta^0_{\mathsf{aux}}(C_0) \odot \mathtt{k_{h}}_i)$ is uniformly in $\mathbb{G}$. The projection key $\overline{\mathtt{k_p}}$ is uniformly at random in $\mathbb{G}^k$ anyway, given the randomness of all $\mathtt{k_{h}}_i$. Note that any violation of $\mathtt{Dec}'_\pi(C_0) = \mathtt{Dec}'_\pi(g(C_1, \ldots, C_x))$ implies the existence of an index j such that $C_j \notin L_{\mathsf{aux}_j}$. $\square$

While smoothness is the foremost property of (extended) smooth projective hash functions, in some cases like password authenticated key exchange pseudorandomness of the produced hash values has to be guaranteed too. Let $\overline{\mathtt{k_p}}$ denote the vector of projection keys $\mathtt{k_{p}}_i$ for $i = 0, \ldots, x$. A SPHF^x is pseudorandom if its hash values are computationally indistinguishable from random without knowledge of the uniformly chosen hash keys $\overline{\mathtt{k_h}}$ or the witnesses $\overline{w}$, i.e. for all $C = (C_0, \ldots, C_x) \in L_{\mathsf{aux}}$ the following distributions are computationally ε-close:

$$\{(\overline{\mathtt{k_p}}, C, h_0, h_x) \mid \forall i \in \{0, \ldots, x\} :\ \mathtt{k_{h}}_i \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathsf{aux}}); \mathtt{k_{p}}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_{h}}_i, L_{\mathsf{aux}}); h_0 \leftarrow \mathtt{Hash}_0(\mathtt{k_{h1}}, \ldots, \mathtt{k_{h}}_x, L_{\mathsf{aux}}, C_0);\ h_x \leftarrow \mathtt{Hash}_x(\mathtt{k_{h0}}, L_{\mathsf{aux}}, C_1, \ldots, C_x)\}$$
$$\overset{\varepsilon}{=} \{(\overline{\mathtt{k_p}}, C, h_0, h_x) \mid \forall i \in \{0, \ldots, x\} :\ \mathtt{k_{h}}_i \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathsf{aux}}); \mathtt{k_{p}}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_{h}}_i, L_{\mathsf{aux}}); h_0 \in_R \mathbb{G}; h_x \in_R \mathbb{G}\}$$

To prove pseudorandomness of an SPHF^x we use modified experiments from [11] given in Definition 5. The proof for the pseudorandomness of SPHF^x follows the line of argument from [11].

Definition 5 (SPHF^x Pseudorandomness). *A SPHF^x Π is pseudorandom if for all PPT algorithms $\mathcal{A}$ there exists a negligible function $\varepsilon(\cdot)$ such that*

$$\mathsf{Adv}^{\mathrm{Pr}}_{\Pi,\mathcal{A}} = \left|\Pr[\mathsf{Exp}^{\mathrm{Pr}}_{\Pi,\mathcal{A}} = 1] - \frac{1}{2}\right| \leq \varepsilon(\lambda)$$

$\mathsf{Exp}^{\mathrm{Pr}}_{\Pi,\mathcal{A}}(\lambda)$: *Choose $b \in_R \{0,1\}$, call $b' \leftarrow \mathcal{A}^{\Omega^{\mathcal{L}}_{\mathsf{pk}}(\cdot)}(\lambda, \mathtt{k_{p}}_0, \ldots, \mathtt{k_{p}}_x)$ with $\mathtt{k_{p}}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_{h}}_i, L_{\mathsf{aux}}, C_i)$ and $\mathtt{k_{h}}_i \leftarrow \mathtt{KGen_H}(L_{\mathsf{aux}})$ for all $i \in 0, \ldots, x$. Return $b = b'$.*

$\Omega^{\mathcal{L}}_{\mathsf{pk}}(\ell, \mathsf{aux})$ *returns elements $C = (C_0, \ldots, C_x) \in L_{\mathsf{aux}}$ with $C_0 \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathsf{pk}}(\ell_0, \mathsf{aux}'; r_0)$ and $C_i \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathsf{pk}}(\ell_i, \mathsf{aux}'_i; r_i)$ for all $i \in 1, \ldots, x$ and $\mathsf{pk} \in \mathsf{aux}$ using encryption scheme $\mathcal{L}$ and according labels ℓ_i. It additionally returns $\mathtt{Hash}_0(\mathtt{k_{h1}}, \ldots, \mathtt{k_{h}}_x, L_{\mathsf{aux}}, C_0), \mathtt{Hash}_x(\mathtt{k_{h0}}, L_{\mathsf{aux}}, C_1, \ldots, C_x)$ if $b = 0$ or $h_0, h_x \in_R \mathbb{G}$ if $b = 1$.*

The following theorem shows pseudorandomness of hash values in SPHF^x.

Theorem 2 (SPHF^x Pseudorandomness). *The SPHF^x construction from Definition 4 on cyclic groups is pseudorandom if $\mathcal{L}$ is a CCA-secure labelled encryption scheme.*

Proof. Pseudorandomness of SPHF^x follows immediately from its smoothness and the CCA-security of the used encryption scheme. First we change $\Omega^{\mathcal{L}}_{\mathsf{pk}}$ such that it returns the encryption of 0 for a random $i \in 0, \dots, x$. This change is not noticeable by the adversary due to the CCA-security of the encryption scheme. Assuming 0 is not a valid message, i.e. $\mathtt{aux}' \neq 0$ and $\mathtt{aux}_i \neq 0$ for all $i \in 1, \dots, x$, the pseudorandomness of SPHF^x follows from its smoothness. $\square$

The authors of [15] furthermore highlight that this definition of pseudorandomness is not enough when used in PAKE protocols if the hash values are not bound to a specific session by signatures or MACs. Therefore, they prove pseudorandomness under re-use of hash keys and ciphertexts. Taking into account re-use of SPHF^x values such as ciphertexts and keys we formalise the notion of concurrent pseudorandomness for SPHF^x following the approach from [15]. Let $\overline{\mathtt{k_p}}$ denote the vector of projection keys $\mathtt{k_p}_i$ for $i = 0, \dots, x$. A SPHF^x is pseudorandom in concurrent execution if the hash values are computationally indistinguishable from random without knowledge of the uniformly chosen hash keys or the witnesses, i.e. for fixed $l = l(\lambda)$ the following distributions are computationally ε-close:

$$\begin{aligned}
&\{(\overline{\mathtt{k_p}_1}, \dots, \overline{\mathtt{k_p}_l}, C_1, \dots, C_l, h_{0,1}, \dots, h_{0,l}, h_{x,1}, \dots, h_{x,l}) \mid \\
&\forall i \in \{0, \dots, x\}, j \in \{1, \dots, l\} : \mathtt{k_h}_{i,j} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathtt{aux}});\ \mathtt{k_p}_{i,j} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}_i, L_{\mathtt{aux}}); \\
&\forall j \in \{1, \dots, l\} : h_{0,j} \leftarrow \mathtt{Hash}_0(\mathtt{k_h}_{1,j}, \dots, \mathtt{k_h}_{x,j}, L_{\mathtt{aux}}, C_{0,j}); \\
&h_{x,j} \leftarrow \mathtt{Hash}_x(\mathtt{k_h}_{0,j}, L_{\mathtt{aux}}, C_{1,j}, \dots, C_{x,j})\} \\
\overset{\varepsilon}{=}\ &\{(\overline{\mathtt{k_p}_1}, \dots, \overline{\mathtt{k_p}_l}, C_1, \dots, C_l, h_{0,1}, \dots, h_{0,l}, h_{x,1}, \dots, h_{x,l}) \mid \\
&\forall i \in \{0, \dots, x\}, j \in \{1, \dots, l\} : \mathtt{k_h}_{i,j} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\mathtt{aux}}); \mathtt{k_p}_{i,j} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}_i, L_{\mathtt{aux}}); \\
&\forall j \in \{1, \dots, l\} : h_{0,j} \in_R \mathbb{G}; h_{x,j} \in_R \mathbb{G}\}
\end{aligned}$$

We extend Definition 5 to capture re-use of hash keys and ciphertexts. The corresponding experiment in Definition 6 generates l hash values to each ciphertext, one for each hash key.

Definition 6 (SPHF^x Concurrent Pseudorandomness). *A SPHF^x Π offers concurrent pseudorandomness if for all PPT algorithms $\mathcal{A}$ and polynomials l there exists a negligible function $\varepsilon(\cdot)$ such that*

$$\mathsf{Adv}^{\mathrm{Pr}}_{\Pi,\mathcal{A}} = \left| \Pr[\mathsf{Exp}^{\mathrm{Pr}}_{\Pi,\mathcal{A}} = 1] - \frac{1}{2} \right| \leq \varepsilon(\lambda)$$

$\mathsf{Exp}^{\mathrm{Pr}}_{\Pi,\mathcal{A}}(\lambda)$: *Choose $b \in_R \{0,1\}$, call $b' \leftarrow \mathcal{A}^{\Omega^{\mathcal{L}}_{\mathsf{pk}}(\cdot)}(\lambda, \overline{\mathtt{k_p}_1}, \dots, \overline{\mathtt{k_p}_l})$ with $\overline{\mathtt{k_p}_j} = (\mathtt{k_p}_0, \dots, \mathtt{k_p}_x)$ where $\mathtt{k_p}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_h}_i, L_{\mathtt{aux}}, C_i)$ and $\mathtt{k_h}_i \leftarrow \mathtt{KGen_H}(L_{\mathtt{aux}})$ for all $i \in 0, \dots, x$ and $j \in 1, \dots, l$. Return $b = b'$.*

$\Omega^{\mathcal{L}}_{\mathsf{pk}}(\ell, \mathtt{aux})$ *returns elements $C = (C_0, \dots, C_x) \in L_{\mathtt{aux}}$ with $C_0 \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathsf{pk}}(\ell_0, \mathtt{aux}'; r_0)$ and $C_i \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathsf{pk}}(\ell_i, \mathtt{aux}_i; r_i)$ for all $i \in 1, \dots, x$ and $\mathsf{pk} \in \mathtt{aux}$ using encryption algorithm $\mathcal{L}$ and according labels ℓ_i. It additionally returns $\mathtt{Hash}_{0,j}(\mathtt{k_h}_{1,j}, \dots, \mathtt{k_h}^j_x, L_{\mathtt{aux}}, C_0), \mathtt{Hash}_{x,j}(\mathtt{k_h}_{0,j}, L_{\mathtt{aux}}, C_1, \dots, C_x)$ if $b = 0$ or $h_{0,j}$, $h_{x,j} \in \mathbb{G}$ if $b = 1$ for all $j \in 1, \dots, l$.*

Using Definition 6 we prove the concurrent pseudorandomness of our construction, following the argument from [15, Lemma 1].

Lemma 1 (SPHF^x Concurrent Pseudorandomness). *The SPHF^x construction from Definition 4 on cyclic groups is pseudorandom on re-use of hash and ciphertext values if $\mathcal{L}$ is a CCA-secure labelled encryption scheme.*

Proof. Using a hybrid argument it is enough to show that the adversary can not distinguish between experiment Exp_1 where Ω returns random elements for the first i hash values of the j-th query and all queries $< j$ and correct hashes for all subsequent queries and indices $> i$, and Exp_2 where Ω returns random elements for the first $i+1$ hash values of the j-th query and all queries $< j$ and correct hashes for all subsequent queries and indices $> i+1$. Having this in mind the proof follows the same argument as the one for SPHF^x pseudorandomness. We briefly recall the argumentation there. We modify Exp_1 to Exp_1' and Exp_2 to Exp_2' such that Ω returns an encryption of 0 instead of correct encryptions for C_j. Note that we assume 0 is not a valid message such that $C_j \notin L_{\mathtt{aux}}$ in Exp_1'. Due to CCA-security of $\mathcal{L}$ this step is not recognisable by the adversary. Changing Exp_1' to Exp_2' the smoothness of SPHF^x ensures that $\mathcal{A}$ can not distinguish between the two experiments, which proves the lemma. $\square$

3.1 Distributed Computation of SPHF^x

Using SPHF^x is only reasonable in a distributed manner. We therefore consider $n = x+1$ entities participating in the distributed computation of the SPHF^x hash values h_0, h_x. Let P_i for $i \in \{1, \ldots, x\}$ denote parties, each knowing $\mathtt{aux}_i$ and computing the according ciphertext C_i and projection key $\mathtt{k_p}_i$. Furthermore, let P_0 denote the participant knowing $\mathtt{aux}$ and computing C_0 and $\mathtt{k_p}_0$. We define protocols in this setting with the purpose that both P_0 and P_1 eventually compute h_0 and h_x.

While P_0 can compute $\mathtt{PHash}_0$ and $\mathtt{Hash}_x$ after receiving all C_i and $\mathtt{k_p}_i$, computation of $\mathtt{Hash}_0$ and $\mathtt{PHash}_x$ can not be performed solely by the previously described algorithms in this setting, without disclosing the witness or the hashing key. To compute $\mathtt{PHash}_x$ and $\mathtt{Hash}_0$, parties $P_1, \ldots, P_x$ have to collaborate since they know only part of the input parameters. Distributed SPHF^x defines protocols that allow secure calculation of h_0 and h_x. Intuitively distributed SPHF^x reaches the same security properties as SPHF^x, namely smoothness and pseudorandomness in presence of a passive adversary, by additionally ensuring that no protocol participant alone is able to compute the hash values. Note that while we assume each P_i for $i > 0$ holds a key-pair and knows public keys of all other P_i such that all communication between two P_i is secured by the receivers public key, those keys are not authenticated, i.e. we do not assume a PKI.

A distributed SPHF^x protocol between n participants $P_0, \ldots, P_x$ computing h_x and h_0 consists of three interactive protocols $\mathtt{Setup}$, $\mathtt{PHash}_x^D$ and $\mathtt{Hash}_0^D$. Let Π denote the SPHF^x algorithm that is being distributed.

- $\mathtt{Setup}(\mathtt{aux}, P_0, \ldots, P_x)$ initialises a new instance for each participant with $(\mathtt{aux}, P_0, P_1, \ldots, P_x)$ for P_0 and $(\mathtt{aux}_i, P_i, P_0, \ldots, P_x)$ for P_i, $i \in \{1, \ldots, x\}$. Eventually, all participants compute and broadcast projection keys $\mathtt{k_p}_i$ and encryptions $C_i \leftarrow \mathtt{Enc}_{\mathtt{pk}}^{\mathcal{L}}(\ell_i, \mathtt{aux}_i'; r_i)$ of their secret $\mathtt{aux}_i'$ using $\Pi.\mathtt{KGen_H}$, $\Pi.\mathtt{KGen_P}$ and the associated encryption scheme $\mathcal{L}$. Participants store incoming $\mathtt{k_p}_i, C_i$ for later use. After receiving $(\mathtt{k_p}_1, C_1, \ldots, \mathtt{k_p}_x, C_x)$, P_0 computes $h_0 \leftarrow \Pi.\mathtt{PHash}_0(\mathtt{k_p}_1, \ldots, \mathtt{k_p}_x, L_{\mathtt{aux}}, C_0, r_0)$ and $h_x \leftarrow \Pi.\mathtt{Hash}_x(\mathtt{k_{h0}}, L_{\mathtt{aux}}, C_1, \ldots, C_x)$.
- $\mathtt{PHash}_x^D$ is executed between parties $P_1, \ldots, P_x$. Each P_i performs $\mathtt{PHash}_x^D$ on input $(\mathtt{k_p}_0, \mathtt{aux}_i, C_1, \ldots, C_x, r_i)$ such that P_1 eventually holds h_x while all P_i for $i > 1$ do not learn anything about h_x.
- $\mathtt{Hash}_0^D$ is executed between parties $P_1, \ldots, P_x$. Each P_i performs $\mathtt{Hash}_0^D$ on input $(\mathtt{aux}_i', \mathtt{k_h}_i, C_0, \ldots, C_x)$ such that P_1 eventually holds h_0 and all P_i for $i > 1$ do not learn anything about h_0.

A distributed SPHF^x is said to be correct if $\mathtt{PHash}_x^D = \mathtt{PHash}_x$ and $\mathtt{Hash}_0^D = \mathtt{Hash}_0$ assuming that all messages are honestly computed and transmitted. The security of the distributed SPHF^x in presence of a passive adversary follows immediately from smoothness and pseudorandomness of the SPHF^x algorithms.

Remark 1. Note that we focus on asymmetric distribution here such that only P_1 computes the hash values. Building symmetric distribution protocols where all parties P_i compute the hash values from this is straightforward but requires a different security model. Likewise, it is possible to build asymmetric distribution protocols where *all* P_i compute *different* hash values (we will see an example of that later).

3.2 Security against Active Adversaries

Smooth projective hashing has not been used in a distributed manner before such that it was not necessary to consider active adversaries. By introducing distributed computation of hash values the $\mathtt{Hash}_0^D$ and

$\mathtt{PHash}_x^D$ protocols are exposed to active attacks. However, the adversary must still not be able to distinguish real hash values from random elements, i.e. smoothness and pseudorandomness must hold. Therefore we introduce a security model for distributed SPHF^x smoothness and pseudorandomness, capturing active attacks in a multi-user and multi-instance setting. Let $\{(P_0^j, P_1^k, \ldots, P_x^l)\}_{P_0^j \in \mathcal{P}_0, P_i^k \in \mathcal{P}\ i\in\{1,\ldots,x\}}$ denote all tuples $(P_0^j, P_1^k, \ldots, P_x^l)$ such that $P_0^j \in \mathcal{P}_0$ knows $\mathtt{aux}$ and $P_1^k, \ldots, P_x^l \in \mathcal{P}$ each know according $\mathtt{aux}_i$. We say P_0 is *registered* with $(P_1, \ldots, P_x)$. The additional indices j, k, l denote the instance of the respective participant (assigned by oracles and modelled as counters to ensure their uniqueness).

Definition 7 (SPHF^x Security). *A distributed SPHF^x protocol Π is secure (offers adaptive smoothness and concurrent pseudorandomness) if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\varepsilon(\cdot)$ such that :*

$$\mathsf{Adv}_{\Pi,\mathcal{A}}^{SPHF^x}(\lambda) = \left|\Pr[\mathsf{Exp}_{\Pi,\mathcal{A}}^{SPHF^x}(\lambda) = 1] - \frac{1}{2}\right| \leq \varepsilon(\lambda)$$

$\mathsf{Exp}_{\Pi,\mathcal{A}}^{SPHF^x}(\lambda)$: *Choose $b \in_R \{0,1\}$, call $b' \leftarrow \mathcal{A}^{\mathsf{Setup}(\cdot),\mathsf{Send}(\cdot),\mathsf{Test}(\cdot)}(\lambda, \mathtt{aux}_2, \ldots, \mathtt{aux}_x, \mathcal{L}, \mathtt{crs})$ and return $b = b'$.*

- $\mathsf{Setup}(P_0, \ldots, P_x)$ *initialises new instances with* $(\mathtt{aux}, P_1, \ldots, P_x)$ *for P_0 registered with* $(P_1, \ldots, P_x)$ *and* $(\mathtt{aux}_1, P_1, P_0, \ldots, P_x)$ *for P_1 and returns* $((\mathtt{k_p}_0, C_0), (\mathtt{k_p}_1, C_1))$ *with* $C_i \leftarrow \mathtt{Enc}_{\mathsf{pk}}^{\mathcal{L}}(\ell, \mathtt{aux}_i'; r_i)$ *and* $\mathtt{k_h}_i \leftarrow \Pi.\mathtt{KGen_H}(L_{\mathtt{aux}})$, $\mathtt{k_p}_i \leftarrow \Pi.\mathtt{KGen_P}(\mathtt{k_h}_i, L_{\mathtt{aux}})$
- $\mathsf{Send}(P_a, P_b, m)$ *sends message m with alleged originator P_b to P_a and returns P_a's resulting message m' if any.*
- $\mathsf{Test}(P_i^j)$ *for $i \in \{0,1\}$ returns two hash values (h_0, h_x). If the global bit b is 0, the hash values are chosen uniformly at random from $\mathbb{G}$, otherwise the hash values are computed according to protocol specification Π.*

Note that we assume without loss of generality that all participants $P_2, \ldots, P_x$ are corrupted by the adversary, who knows their secrets. Furthermore, note that $\mathcal{A}$ can query the Test oracle only once.

The active security notion for distributed computation of SPHF^x covers smoothness and pseudorandomess as defined before. The experiment is equivalent to the computational smoothness definition when $\mathcal{A}$ computes and forwards all messages honestly but changes at least one $\mathtt{aux}_i$. Note that this is actually a stronger notion than smoothness as we require pseudorandomness of hash values output by the projection function on a word not in the language. This is usually not included in the smoothness definition, which is defined over the hash function. Further, Definition 7 is equivalent to Definition 6 when $\mathcal{A}$ computes and forwards all messages honestly and does *not* change any $\mathtt{aux}_i$.

3.3 Instantiation – Distributed Cramer-Shoup SPHF^x

We exemplify the SPHF^x definition on the previously introduced Cramer-Shoup encryption scheme (a second instantiation for ElGamal ciphertexts can be found in Appendix A.3). The ciphertexts are created as $C_i = (u_{1,i}, u_{2,i}, e_i, v_i) \leftarrow \mathtt{Enc}_{\mathsf{pk}}^{\mathsf{CS}}(\ell_i, \mathtt{aux}_i'; r_i)$ for all $i = 1, \ldots, x$ with $\mathtt{aux}_i' = h(\mathtt{aux}')[i]$ and $C_0 = (u_{1,0}, u_{2,0}, e_0, v_0) \leftarrow \mathtt{Enc}_{\mathsf{pk}}^{\mathsf{CS}}(\ell_0, \mathtt{aux}_0'; r_0)$, where ℓ_i consists of participating parties and the party's projection key. We define modified decryption as $\mathtt{Dec}_\pi'(C) = e \cdot u_1^{-z}$. The combining function g uses the homomorphic property of u_1 and e of the CS ciphertext such that $g(C_1, \ldots, C_x) = (\prod_{i=1}^x u_{1,i}, \prod_{i=1}^x e_i)$ and $\mathtt{aux}' = \sum_{i=1}^x \mathtt{aux}_i'$. The following variables define the Cramer-Shoup SPHF^x:

$$\Gamma = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \in \mathbb{G}^{2\times 5}, \quad \lambda = (r, r\xi) \in \mathbb{Z}_p^{1\times 2}$$

$$\Theta_{\mathtt{aux}}^0(C_0) = (u_1, u_1^\xi, u_2, e/\mathtt{aux}', v) \in \mathbb{G}^{1\times 5}$$

$$\Theta_{\mathtt{aux}}^x(C_1, \ldots, C_x) = (\prod_{i=1}^x u_{1,i}, \prod_{i=1}^x u_{1,i}^{\xi_i}, \prod_{i=1}^x u_{2,i}, \prod_{i=1}^x e_i/\mathtt{aux}', \prod_{i=1}^x v_i) \in \mathbb{G}^{1\times 5}$$

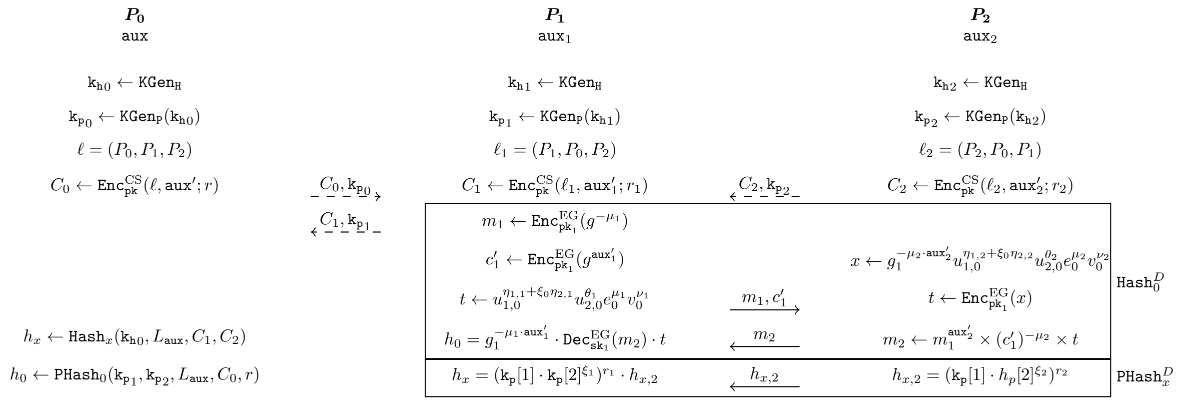
Using them in the SPHF^x Definition 4 yields the Cramer-Shoup SPHF^x. For a detailed description of the resulting SPHF^x see Appendix A.2. Instead of aiming for absolute generality we describe the distributed Cramer-Shoup SPHF^x for $x = 2$ such that both participants P_1 and P_2 compute and broadcast $(\mathtt{k_{p}}_i, C_i)$, while P_0 computes and broadcasts $(\mathtt{k_{p0}}, C_0)$. Let $\times$ denote element wise multiplication, e.g., for El-Gamal ciphertexts $C_1 = (u_1, e_1), C_2 = (u_2, e_2)$, $C_1 \times C_2$ is defined as (u_1u_2, e_1e_2). $\mathtt{PHash}_x^D$ and $\mathtt{Hash}_0^D$ protocols are defined as follows (Figure 1 depicts the entire SPHF^x execution):

– $\mathtt{PHash}_x^D$ is executed between P_1 and P_2. P_2 computes $h_{x,2} = \lambda \odot \mathtt{k_{p0}} = (\mathtt{k_{p0}}[1] \cdot \mathtt{k_{p0}}[2]^{\xi_2})^{r_2}$ and sends it to P_1. Eventually, P_1 holds $h_x = h_{x,2} \cdot (\lambda \odot \mathtt{k_{p0}}) = \mathtt{k_{p0}}[1]^{r_1+r_2} \cdot \mathtt{k_{p0}}[2]^{\xi_1 \cdot r_1 + \xi_2 \cdot r_2}$. Note that P_1 always performs checks that $\mathtt{k_{p0}} \in \mathbb{G}$ and $\mathbb{G} \ni h_2^x \neq 0$.
– $\mathtt{Hash}_0^D$ is executed between P_1 and P_2 such that P_1 eventually holds h_0. Let P_i for $i \in \{1,2\}$ denote the participating party knowing $(\mathtt{aux}_i, \mathtt{sk}_i, \mathtt{k_{h}}_i = (\eta_1, \eta_2, \theta, \mu, \nu), \mathtt{pk}_1, \mathtt{pk}_2, C_0 = (u_1, u_2, e, v, \xi))$.
  • P_1 computes $m_0 \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_1}(g_1^{-\mu}; r)$ and $c'_1 \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_1}(g_1^{\mathtt{aux}'_1}; r')$, and sends (m_0, c'_1) to P_2.
  • Receiving (m_0, c'_1) from P_1, P_2 computes

$$m_1 \leftarrow (m_0)^{\mathtt{aux}'_2} \times (c'_1)^{-\mu} \times \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_1}(g_1^{-\mu \cdot \mathtt{aux}'_2} \cdot u_1^{\eta_1 + \xi\eta_2} \cdot u_2^{\theta} \cdot e^{\mu} \cdot v^{\nu}; r'')$$

  and sends it to P_1.
  • Receiving m_1, P_1 computes the hash value

$$h_0 = g_1^{-\mu \cdot \mathtt{aux}'_1} \cdot \mathtt{Dec}^{\mathrm{EG}}_{\mathtt{sk}_1}(m_1) \cdot u_1^{\eta_1 + \xi\eta_2} \cdot u_2^{\theta} \cdot e^{\mu} \cdot v^{\nu}.$$



P_0
aux
P_1
$\mathtt{aux}_1$
P_2
$\mathtt{aux}_2$
$\mathtt{k_{h0}} \leftarrow \mathtt{KGen_H}$
$\mathtt{k_{h1}} \leftarrow \mathtt{KGen_H}$
$\mathtt{k_{h2}} \leftarrow \mathtt{KGen_H}$
$\mathtt{k_{p0}} \leftarrow \mathtt{KGen_P}(\mathtt{k_{h0}})$
$\mathtt{k_{p1}} \leftarrow \mathtt{KGen_P}(\mathtt{k_{h1}})$
$\mathtt{k_{p2}} \leftarrow \mathtt{KGen_P}(\mathtt{k_{h2}})$
$\ell = (P_0, P_1, P_2)$
$\ell_1 = (P_1, P_0, P_2)$
$\ell_2 = (P_2, P_0, P_1)$
$C_0 \leftarrow \mathtt{Enc}^{\mathrm{CS}}_{\mathtt{pk}}(\ell, \mathtt{aux}'; r)$
$C_0, \mathtt{k_{p0}}$
$C_1 \leftarrow \mathtt{Enc}^{\mathrm{CS}}_{\mathtt{pk}}(\ell_1, \mathtt{aux}'_1; r_1)$
$C_2, \mathtt{k_{p2}}$
$C_2 \leftarrow \mathtt{Enc}^{\mathrm{CS}}_{\mathtt{pk}}(\ell_2, \mathtt{aux}'_2; r_2)$
$C_1, \mathtt{k_{p1}}$
$m_1 \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_1}(g^{-\mu_1})$
$c'_1 \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_1}(g^{\mathtt{aux}'_1})$
$x \leftarrow g_1^{-\mu_2 \cdot \mathtt{aux}'_2} u_{1,0}^{\eta_{1,2}+\xi_0\eta_{2,2}} u_{2,0}^{\theta_2} e_0^{\mu_2} v_0^{\nu_2}$
$\mathtt{Hash}_0^D$
$t \leftarrow u_{1,0}^{\eta_{1,1}+\xi_0\eta_{2,1}} u_{2,0}^{\theta_1} e_0^{\mu_1} v_0^{\nu_1}$
m_1, c'_1
$t \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_1}(x)$
$h_x \leftarrow \mathtt{Hash}_x(\mathtt{k_{h0}}, L_{\mathtt{aux}}, C_1, C_2)$
$h_0 = g_1^{-\mu_1 \cdot \mathtt{aux}'_1} \cdot \mathtt{Dec}^{\mathrm{EG}}_{\mathtt{sk}_1}(m_2) \cdot t$
m_2
$m_2 \leftarrow m_1^{\mathtt{aux}'_2} \times (c'_1)^{-\mu_2} \times t$
$h_0 \leftarrow \mathtt{PHash}_0(\mathtt{k_{p1}}, \mathtt{k_{p2}}, L_{\mathtt{aux}}, C_0, r)$
$h_x = (\mathtt{k_p}[1] \cdot \mathtt{k_p}[2]^{\xi_1})^{r_1} \cdot h_{x,2}$
$h_{x,2}$
$h_{x,2} = (\mathtt{k_p}[1] \cdot h_p[2]^{\xi_2})^{r_2}$
$\mathtt{PHash}_x^D$

Fig. 1: Distributed Cramer-Shoup SPHF^x

Dashed lines denote broadcast messages.

Security of Distributed Cramer-Shoup SPHF^x We show now that the proposed distributed Cramer-Shoup SPHF^x is secure. The intuition behind the proof is that the pseudorandomness of h_x can be reduced directly to the DDH problem in $\mathbb{G}$ while pseudorandomness of h_0 value follows from the smoothness and pseudorandomness of the underlying SPHF^x scheme.

Theorem 3 (Cramer-Shoup SPHF^x Security). *The distributed Cramer-Shoup SPHF^x instantiation is secure against active adversaries according to Definition 7 when the DDH assumption in the used group $\mathbb{G}$ holds and $\mathcal{L} = \mathrm{CS}$ is CCA-secure.*

Proof. First, note that the theorem follows immediately from smoothness and pseudorandomness in the passive case if the adversary queries $\mathsf{Test}(P_0)$. We therefore focus on $\mathsf{Test}(P_1)$ queries. We start with the pseudorandomness of h_x, i.e. for all g it holds that $\Pr[h_x = g] = 1/|\mathbb{G}|$. Consider an attacker $\mathcal{A}$ on input $(\lambda, \mathtt{aux}_2, \mathcal{L}, \mathtt{crs})$ and let Exp_0 denote the original SPHF^x experiment.

$\mathsf{Exp_1}$: We change Test such that a uniformly at random chosen element $g_x \in_R \mathbb{G}$ is returned for h_x.

Claim. $\left|\mathsf{Adv}^{\mathsf{Exp}_0}_{\Pi,\mathcal{A}} - \mathsf{Adv}^{\mathsf{Exp}_1}_{\Pi,\mathcal{A}}\right| \leq \varepsilon(\lambda)$

Proof. The hash value h_x in Exp_0 is computed as $h_x = (\mathtt{k_p}{}_0'[1] \cdot \mathtt{k_p}{}_0'[2]^{\xi_1})^{r_1} \cdot h_{x,2}$ with adversarially generated $h_{x,2}$ and $\mathtt{k_p}{}_0'$. Indistinguishability of h_x and g_x, and thus the claim, follows immediately as long as the DDH assumption in $\mathbb{G}$ holds (using DDH triple $(\mathtt{k_p}{}_0'[1] \cdot \mathtt{k_p}{}_0'[2]^{\xi_1}, g^{r_1}, h_x)$ and $(\mathtt{k_p}{}_0'[1] \cdot \mathtt{k_p}{}_0'[2]^{\xi_1}, g^{r_1}, g_x)$). Note that P_1 aborts if either $h_{x,2} \notin \mathbb{G}$ or $\mathtt{k_p}{}_0' \notin \mathbb{G}^2$. □

To show the security (concurrent pseudorandomness and adaptive smoothness) of h_0 we define two Send queries that allow execution of the protocol: $(m_1, c_1') \leftarrow \mathsf{Send}_1(P_2, P_1, (\mathtt{k_p}{}_0', C_0', \mathtt{k_p}{}_2', C_2'))$ starts the protocol execution between P_1 and P_2 and provides the attacker with (m_1, c_1'). Using these messages the adversary (P_2) computes a message m_2 and sends it to P_1 with $\mathsf{Send}_2(P_2, P_1, m_2)$. This reflects the execution of a single protocol run of $\mathtt{Hash}_0^D$ such that P_1 eventually computes h_0. In contrast to the passive and classical SPHF proofs we can not replace the ciphertexts with encryptions of words not in the language. However, this is not necessary as t is in fact the $\mathtt{Hash}$ computation of the classical Cramer-Shoup SPHF without cancelling the message, i.e. $t = h \cdot m^{\mu}$.

$\mathsf{Exp_2}$: We change Test such that a uniformly at random chosen element $g_0 \in_R \mathbb{G}$ is returned for h_0.

Claim. $\left|\mathsf{Adv}^{\mathsf{Exp}_1}_{\Pi,\mathcal{A}} - \mathsf{Adv}^{\mathsf{Exp}_2}_{\Pi,\mathcal{A}}\right| \leq \varepsilon(\lambda)$

Proof. The hash value h_0 in Exp_1 is computed as $h_0 = g^{-\mu_1 \cdot \mathtt{aux}_1'} \cdot \mathsf{Dec}^{\mathrm{EG}}_{\mathtt{sk}_1}(m_2) \cdot t$ with $t = u_{1,0}^{\eta_{1,1}+\xi_0\eta_{2,1}} u_{2,0}^{\theta_1} e_0^{\mu_1} v_0^{\nu_1}$ where m_2 and $C_0' = (u_{1,0}, u_{2,0}, e_0, v_0)$ may be adversarially generated. The value t is actually the $\mathtt{Hash}$ value of the classical Cramer-Shoup SPHF without cancelled message, or in other words t is the result of a SPHF $\mathtt{Hash}$ computation for language $L_{(\mathtt{crs},0)}$ such that any C_0', encrypting some correct $\mathtt{aux}' \neq 0$, is not in this language. Due to smoothness of the $\mathtt{Hash}$ function [6] t is indistinguishable from a uniformly at random chosen element. If the adversary encrypted 0 in C_0' pseudorandomness of $\mathtt{Hash}$ takes effect. Therefore $h_0 = d \cdot t$ is indistinguishable from a random group element for all $d \in \mathbb{G}$. □

In Exp_2 the adversary always gets random group elements in answer to his Test query. Therefore, he can not do better than guessing bit b. □

4 Two-Server PAKE from Distributed SPHF^x

In this section we present a new two-server PAKE framework as an application of our distributed SPHF^x concept. Moreover, we show that the two-server PAKE protocol by Katz et al. [13] can be considered as a variant of our framework using a "mix" of distributed SPHF^x for Cramer-Shoup and El-Gamal ciphertexts.

With a single server storing the password, password authenticated key exchange (PAKE) protocols have an intrinsic single point of failure. As soon as the server's database, storing the client's secrets, gets compromised the attacker can impersonate the client to this server, and most likely also to others considering that users tend to reuse their passwords across multiple services. Mechanisms have been proposed to solve the problem of server compromise [12,19]. However, as long as only one server is used, PAKE protocols are prone to offline dictionary attacks on the server side. Two-server PAKE (2PAKE) protocols can solve this problem by splitting the password in two parts such that a malicious or compromised server can be used to recover only one part of the password. Raimondo and Gennaro [17] proposed a t-out-of-n threshold PAKE, which is not suitable for the 2PAKE setting as it requires $t < n/3$. Another t-out-of-n threshold PAKE was proposed in a PKI-based setting with random oracles [16]. Brainard and Juels [8] proposed two-server password based authentication without security proof. Szydlo and Kaliski [18] later modified constuctions from [8] and proved their security in a simulation-based model. The first two-server PAKE in the password-only setting, i.e. without a PKI, is due to Katz et al. [13], based on the KOY protocol from [14]. We consider the same setting as [13] in which the client computes two independent session keys with the two servers.

4.1 A new Two-Server PAKE Framework

Using distributed SPHF^x we can build efficient 2PAKE protocols. We consider the same setting as 2KOY [13], in particular a client that negotiates independent session keys with both servers that hold $\mathrm{pw}_1 + \mathrm{pw}_2 = \mathrm{pw}$. We omit the second server in the description of the protocol in Figure 2 as the framework is symmetric in the sense that the second server S_2 performs like S_1. The framework follows the same principle as the latest PAKE frameworks from SPHFs. In particular it can be seen as a two-server variant of the PAKE protocol from [15].

You can think of the two-server protocol as the execution of two distributed SPHF^x protocols, one between (C, S_1, S_2) and one between (C, S_2, S_1) where servers S_2 and S_1 swap roles, such that (C, S_1) and (C, S_2) eventually hold common hash values that can be used to generate a shared session key $\mathtt{sk}_1$ and $\mathtt{sk}_2$. The only overlap between the two SPHF^x executions is the $\mathtt{Hash}_x$ computation. The reuse of C_1, C_2 in $\mathtt{Hash}_x$ functions is covered by the concurrent pseudorandomness.

2PAKE Framework The servers encrypt their password shares under a public key $\mathtt{pk}$ stored in the $\mathtt{crs}$ using a CCA-secure labelled encryption scheme and distribute this ciphertext together with two appropriate projection keys for a secure distributed SPHF^x, $(\mathtt{k}_{\mathtt{p}_{1,1}}, \mathtt{k}_{\mathtt{p}_{1,2}}, C_1)$ and $(\mathtt{k}_{\mathtt{p}_{2,1}}, \mathtt{k}_{\mathtt{p}_{2,2}}, C_2)$. The client computes two independent encryptions of the password and generates two independent according projection keys $(\mathtt{k}_{\mathtt{p}_{0,1}}, C_{0,1}, \mathtt{k}_{\mathtt{p}_{0,2}}, C_{0,2})$. The previously described SPHF^x allows us to send all $\mathtt{k}_{\mathtt{p}_i}, C_i$ in one round and therefore reach optimality for this step. Using these values, the client can compute session keys as product of the two hash values $h_{0,1}, h_{x,1}$ for $\mathtt{sk}_1$, which is shared with S_1 and from $h_{0,2}, h_{x,2}$ for $\mathtt{sk}_2$ that is shared with S_2.

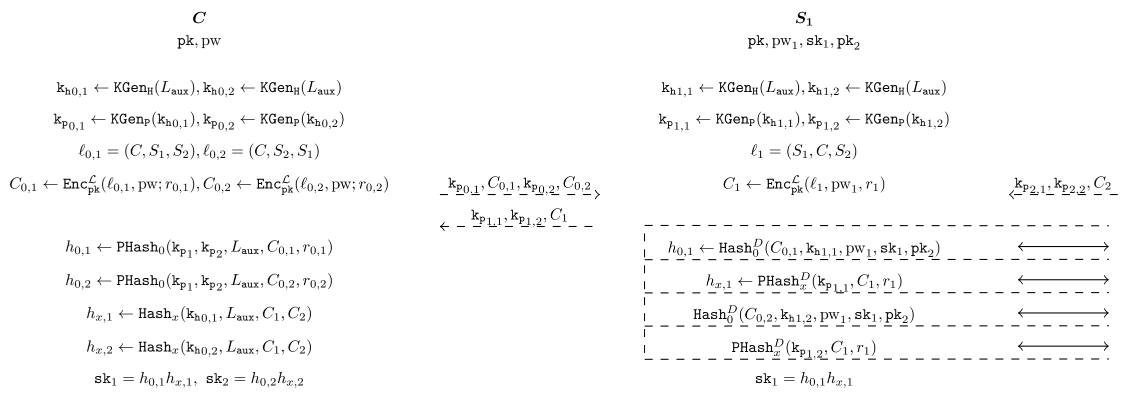
Subsequently, the two servers perform the $\mathtt{Hash}_0^D$ and $\mathtt{PHash}_x^D$ protocols such that S_1 and S_2 eventually hold hash values $h_{0,1}$ and $h_{x,1}$, $h_{0,2}$ and $h_{x,2}$ respectively, to compute $\mathtt{sk}_1$, $\mathtt{sk}_2$ respectively. Eventually, C holds $\mathtt{sk}_1 = h_{0,1} \cdot h_{x,1}$ and $\mathtt{sk}_2 = h_{0,2} \cdot h_{x,2}$, S_1 holds $\mathtt{sk}_1 = h_{0,1} \cdot h_{x,1}$ and S_2 holds $\mathtt{sk}_2 = h_{0,2} \cdot h_{x,2}$. An instantiation of the framework using labelled Cramer-Shoup encryption and the aforementioned distributed SPHF^x yields a secure 2PAKE protocol. Note that this actually requires two SPHF^x executions.

Security We use the well-known game based PAKE model first introduced by Bellare et al. [3] in it's two-server variant from [13]. For a formal description of the model we refer to [13]. The security of the two-server PAKE framework follows directly from the CCA-security of the used encryption scheme and the security of the distributed SPHF^x.

Theorem 4. *Let* $(\mathtt{KGen}_{\mathtt{H}}, \mathtt{KGen}_{\mathtt{P}}, \mathtt{PHash}_0, \mathtt{Hash}_x, \mathtt{Hash}_0^D, \mathtt{PHash}_x^D)$ *be a secure distributed SPHF*^{x} *and* ($\mathtt{KGen}$, $\mathtt{Enc}$, $\mathtt{Dec}$) *a CCA-secure labelled encryption scheme, then the proposed framework in Figure 2 is a secure two-server PAKE protocol.*

Proof (sketch). Let Π denote a secure instantiation of the 2PAKE framework. To prove security of Π we introduce three experiments such that the adversary in the last experiment Exp_3 can not do better than guessing the password as all messages are password independent, i.e. $\mathsf{Adv}_{\Pi,\mathcal{A}}^{\mathsf{Exp}_3} \leq q/|\mathcal{D}|$ for q active attacks. We initially focus on the AKE-security of $\mathtt{sk}_1$.

Exp_1 is identical to the two-server AKE-security experiment except that the simulator knows π, the decryption key to $\mathtt{pk}$ in the $\mathtt{crs}$ (only a syntactical change) and the following changes: If $C_{0,1}$ or C_1, handed to S_1 or C are adversarially generated and encrypt the correct password(share), the simulator stops and $\mathcal{A}$



C
pk, pw
$\mathtt{k}_{\mathtt{h}0,1} \leftarrow \mathtt{KGen}_{\mathtt{H}}(L_{\mathsf{aux}}), \mathtt{k}_{\mathtt{h}0,2} \leftarrow \mathtt{KGen}_{\mathtt{H}}(L_{\mathsf{aux}})$
$\mathtt{k}_{\mathtt{P}0,1} \leftarrow \mathtt{KGen}_{\mathtt{P}}(\mathtt{k}_{\mathtt{h}0,1}), \mathtt{k}_{\mathtt{P}0,2} \leftarrow \mathtt{KGen}_{\mathtt{P}}(\mathtt{k}_{\mathtt{h}0,2})$
$\ell_{0,1} = (C, S_1, S_2), \ell_{0,2} = (C, S_2, S_1)$
$C_{0,1} \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\ell_{0,1}, \mathrm{pw}; r_{0,1}), C_{0,2} \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\ell_{0,2}, \mathrm{pw}; r_{0,2})$
$h_{0,1} \leftarrow \mathtt{PHash}_0(\mathtt{k}_{\mathtt{P}_1}, \mathtt{k}_{\mathtt{P}_2}, L_{\mathsf{aux}}, C_{0,1}, r_{0,1})$
$h_{0,2} \leftarrow \mathtt{PHash}_0(\mathtt{k}_{\mathtt{P}_1}, \mathtt{k}_{\mathtt{P}_2}, L_{\mathsf{aux}}, C_{0,2}, r_{0,2})$
$h_{x,1} \leftarrow \mathtt{Hash}_x(\mathtt{k}_{\mathtt{h}0,1}, L_{\mathsf{aux}}, C_1, C_2)$
$h_{x,2} \leftarrow \mathtt{Hash}_x(\mathtt{k}_{\mathtt{h}0,2}, L_{\mathsf{aux}}, C_1, C_2)$
$\mathtt{sk}_1 = h_{0,1}h_{x,1},\ \mathtt{sk}_2 = h_{0,2}h_{x,2}$
$\mathtt{k}_{\mathtt{P}0,1}, C_{0,1}, \mathtt{k}_{\mathtt{P}0,2}, C_{0,2}$
$\mathtt{k}_{\mathtt{P}1,1}, \mathtt{k}_{\mathtt{P}1,2}, C_1$
S_1
$\mathrm{pk}, \mathrm{pw}_1, \mathrm{sk}_1, \mathrm{pk}_2$
$\mathtt{k}_{\mathtt{h}1,1} \leftarrow \mathtt{KGen}_{\mathtt{H}}(L_{\mathsf{aux}}), \mathtt{k}_{\mathtt{h}1,2} \leftarrow \mathtt{KGen}_{\mathtt{H}}(L_{\mathsf{aux}})$
$\mathtt{k}_{\mathtt{P}1,1} \leftarrow \mathtt{KGen}_{\mathtt{P}}(\mathtt{k}_{\mathtt{h}1,1}), \mathtt{k}_{\mathtt{P}1,2} \leftarrow \mathtt{KGen}_{\mathtt{P}}(\mathtt{k}_{\mathtt{h}1,2})$
$\ell_1 = (S_1, C, S_2)$
$C_1 \leftarrow \mathtt{Enc}^{\mathcal{L}}_{\mathtt{pk}}(\ell_1, \mathrm{pw}_1, r_1)$
$\mathtt{k}_{\mathtt{P}2,1}, \mathtt{k}_{\mathtt{P}2,2}, C_2$
$h_{0,1} \leftarrow \mathtt{Hash}^D_0(C_{0,1}, \mathtt{k}_{\mathtt{h}1,1}, \mathrm{pw}_1, \mathtt{sk}_1, \mathtt{pk}_2)$
$h_{x,1} \leftarrow \mathtt{PHash}^D_x(\mathtt{k}_{\mathtt{P}1,1}, C_1, r_1)$
$\mathtt{Hash}^D_0(C_{0,2}, \mathtt{k}_{\mathtt{h}1,2}, \mathrm{pw}_1, \mathtt{sk}_1, \mathtt{pk}_2)$
$\mathtt{PHash}^D_x(\mathtt{k}_{\mathtt{P}1,2}, C_1, r_1)$
$\mathtt{sk}_1 = h_{0,1}h_{x,1}$

Fig. 2: Two-Server PAKE framework using SPHF^x

Dashed lines denote broadcast messages.

wins the experiment. If $C_{0,1}$, C_1 or C_2, handed to S_1 or C encrypt a wrong password(share), the key for that session is drawn uniformly at random from $\mathbb{G}$. The first change only increases the adversarial advantage and the second one introduces a negligible gap according to the adaptive smoothness of the used SPHF^x.

Exp_2 performs like Exp_1 except that it draws the session key at random from $\mathbb{G}$ if all C_i handed to C and S_1 are oracle generated or encrypt the correct password and no session key has been chosen for the partner in that session (otherwise that previously drawn key is used). This introduces a negligible gap between advantages in Exp_1 and Exp_2 due to the concurrent pseudorandomness of the used SPHF^x.

Exp_3 acts like Exp_2 except that it returns encryptions of 0 for $C_{0,1}$ and C_1 (note that 0 is not a valid password). This step is covered by the CCA-security of the used encryption scheme.

AKE-security of $\mathtt{sk}_1$ follows as all messages are password independent in Exp_3 unless the adversary guesses the correct password. Using the same sequence of experiments but considering C and S_2 instead of C and S_1, AKE-security of $\mathtt{sk}_2$ follows. □

4.2 2-Server KOY (2KOY) [13]

We can now "explain" the use of SPHF in 2KOY from [13]; similar to [11] that "explained" the original KOY protocol from [14]. We define encryption schemes and distributed SPHF^x used in 2KOY, highlight changes to our framework and discuss implications of this on the security of 2KOY.

The $\mathtt{crs}$ contains a public key $\mathtt{pk}$ for Cramer-Shoup encryption as well as a public key g_3 for El-Gamal encryption. Since [13] uses El-Gamal encryptions on the server side, we have to use a combination of Cramer-Shoup and El-Gamal based SPHF^x in 2KOY. Instead of using Cramer-Shoup encryptions and SPHF^x, the client computes projection keys for an El-Gamal distributed SPHF^x, which is based on the aforementioned SPHF on El-Gamal ciphertexts.

Likewise, the servers compute projection keys for a Cramer-Shoup distributed GL-SPHF^x and El-Gamal encryptions of their password shares.[2] We describe the original GL-SPHF on Cramer-Shoup ciphertexts in Appendix A.1 The client sends the projection keys in a third round together with a signature on the session transcript to the servers. The protocol is depicted in Figure 3. Note that we moved K^r into a separate encryption compared to the original protocol. The ElGamal encryption of the password of party i, $\hat{C}_i \leftarrow \mathtt{Enc}^{\mathrm{EG}}_{\mathtt{pk}_i}(g^{\mathrm{pw}_i}; r_i)$ is precomputed and stored on $S_j, j \neq i, j \in \{1,2\}$. Eventually, the client computes hash values using the $\mathtt{PHash}_0$ function of the GL-SPHF^x scheme on CS ciphertexts and the $\mathtt{Hash}_x$ function of the SPHF^x scheme on El-Gamal ciphertexts. Further, the servers execute the $\mathtt{Hash}^D_0$ protocol of the

[2]Note that an additional signature on the session transcript in round three ensures "non-malleability" of these ciphertexts.

distributed GL-SPHF^x scheme on CS ciphertexts and the $\texttt{PHash}_x^D$ protocol of the distributed SPHF^x scheme on El-Gamal ciphertexts.

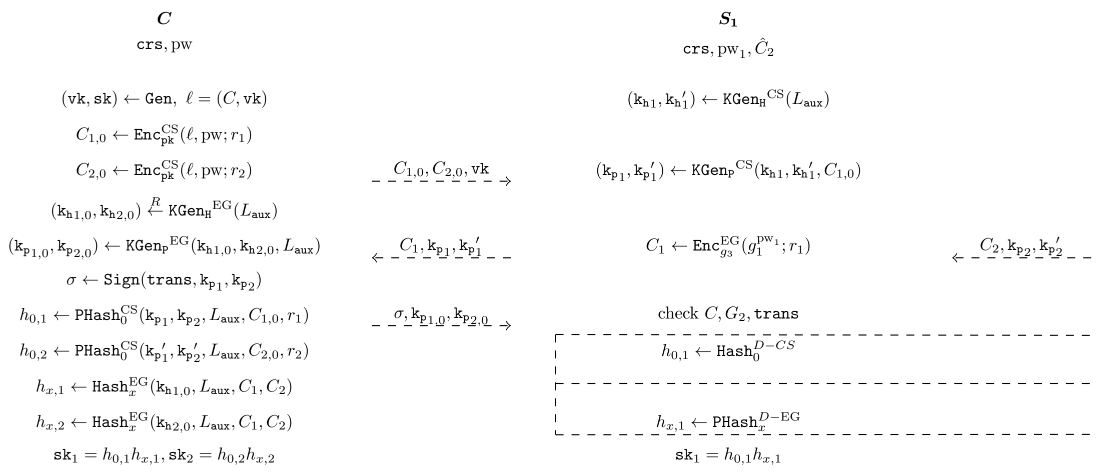

C
$\texttt{crs}, \mathrm{pw}$
S_1
$\texttt{crs}, \mathrm{pw}_1, \hat{C}_2$
$(\texttt{vk}, \texttt{sk}) \leftarrow \texttt{Gen},\ \ell = (C, \texttt{vk})$
$(\texttt{k}_{\texttt{h}1}, \texttt{k}_{\texttt{h}1}') \leftarrow \texttt{KGen}_{\texttt{H}}{}^{\text{CS}}(L_{\text{aux}})$
$C_{1,0} \leftarrow \texttt{Enc}_{\texttt{pk}}^{\text{CS}}(\ell, \mathrm{pw}; r_1)$
$C_{2,0} \leftarrow \texttt{Enc}_{\texttt{pk}}^{\text{CS}}(\ell, \mathrm{pw}; r_2)$
$C_{1,0}, C_{2,0}, \texttt{vk}$
$(\texttt{k}_{\texttt{p}1}, \texttt{k}_{\texttt{p}1}') \leftarrow \texttt{KGen}_{\texttt{P}}{}^{\text{CS}}(\texttt{k}_{\texttt{h}1}, \texttt{k}_{\texttt{h}1}', C_{1,0})$
$(\texttt{k}_{\texttt{h}1,0}, \texttt{k}_{\texttt{h}2,0}) \overset{R}{\leftarrow} \texttt{KGen}_{\texttt{H}}{}^{\text{EG}}(L_{\text{aux}})$
$(\texttt{k}_{\texttt{p}1,0}, \texttt{k}_{\texttt{p}2,0}) \leftarrow \texttt{KGen}_{\texttt{P}}{}^{\text{EG}}(\texttt{k}_{\texttt{h}1,0}, \texttt{k}_{\texttt{h}2,0}, L_{\text{aux}})$
$C_1, \texttt{k}_{\texttt{p}1}, \texttt{k}_{\texttt{p}1}'$
$C_1 \leftarrow \texttt{Enc}_{g_3}^{\text{EG}}(g_1^{\mathrm{pw}_1}; r_1)$
$C_2, \texttt{k}_{\texttt{p}2}, \texttt{k}_{\texttt{p}2}'$
$\sigma \leftarrow \texttt{Sign}(\texttt{trans}, \texttt{k}_{\texttt{p}1}, \texttt{k}_{\texttt{p}2})$
$h_{0,1} \leftarrow \texttt{PHash}_0^{\text{CS}}(\texttt{k}_{\texttt{p}1}, \texttt{k}_{\texttt{p}2}, L_{\text{aux}}, C_{1,0}, r_1)$
$\sigma, \texttt{k}_{\texttt{p}1,0}, \texttt{k}_{\texttt{p}2,0}$
check $C, G_2, \texttt{trans}$
$h_{0,2} \leftarrow \texttt{PHash}_0^{\text{CS}}(\texttt{k}_{\texttt{p}1}', \texttt{k}_{\texttt{p}2}', L_{\text{aux}}, C_{2,0}, r_2)$
$h_{0,1} \leftarrow \texttt{Hash}_0^{D-CS}$
$h_{x,1} \leftarrow \texttt{Hash}_x^{\text{EG}}(\texttt{k}_{\texttt{h}1,0}, L_{\text{aux}}, C_1, C_2)$
$h_{x,2} \leftarrow \texttt{Hash}_x^{\text{EG}}(\texttt{k}_{\texttt{h}2,0}, L_{\text{aux}}, C_1, C_2)$
$h_{x,1} \leftarrow \texttt{PHash}_x^{D-\text{EG}}$
$\texttt{sk}_1 = h_{0,1} h_{x,1}, \texttt{sk}_2 = h_{0,2} h_{x,2}$
$\texttt{sk}_1 = h_{0,1} h_{x,1}$

Fig. 3: Two-Server KOY [13] using SPHF^x

Dashed lines denote broadcast messages.

Security of 2KOY Security of the protocol in Figure 3 against passive adversaries follows immediately from [13, Theorem 1] as we do not change the protocol. However, the authors of [13] need additional mechanisms to prove their protocol secure against an active adversary. They add witness-indistinguishable Σ-protocols to the $\texttt{PHash}_x^D$ and $\texttt{Hash}_0^D$ protocols that prove correctness of their messages. Without giving a proof it should be clear that Theorem 4 also holds for the 2KOY instantiation *without* additional mechanisms. Examining the proof of [13, Theorem 2] shows that the additional steps are only necessary to conduct the proof without actually giving additional security. This shows the power of distributed SPHF^x as they allow for much simpler proofs of multi-party protocols. Furthermore, with our framework the protocol becomes more efficient than 2KOY as it needs only two rounds instead of three and does not need correctness proofs in the distributed hash and projection protocols.

5 Conclusion

We introduced the notion of extended (distributed) smooth projective hashing and gave an instantiation using Cramer-Shoup ciphertexts. Distributed smooth projective hashing can be used as building block in threshold and multi-party protocols. As an example, we built a two-server PAKE framework using a distributed smooth projective hash function. This two-server PAKE framework yields the most efficient two-server PAKE protocols today. The framework also allows us to explain and simplify the two-server PAKE protocol from [13].

While we focused on two-server password authenticated key exchange as application of distributed SPHF in this work, (distributed) extended smooth projective hash functions is an interesting building block for future work on other multi-party and threshold protocols.

Acknowledgments This research was supported by the German Science Foundation (DFG) through the project PRIMAKE (MA 4957)

References

1. M. Abdalla, C. Chevalier, and D. Pointcheval. Smooth Projective Hashing for Conditionally Extractable Commitments. In *CRYPTO'09*, volume 5677 of *Lecture Notes in Computer Science*, pages 671–689. Springer-Verlag, 2009. 1
2. M. Abdalla and D. Pointcheval. A Scalable Password-Based Group Key Exchange Protocol in the Standard Model. In *ASIACRYPT'06*, volume 4284 of *Lecture Notes in Computer Science*, pages 332–347. Springer-Verlag, 2006. 1
3. M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT'00*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 2000. 11
4. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In *PKC'13*, volume 7778 of *Lecture Notes in Computer Science*, pages 272–291. Springer-Verlag, 2013. 1
5. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New Smooth Projective Hash Functions and One-Round Authenticated Key Exchange. Cryptology ePrint Archive, Report 2013/034, 2013. `http://eprint.iacr.org/`. 1, 2, 3, 4, 5, 15
6. F. Benhamouda, O. Blazy, C. Chevalier, D. Pointcheval, and D. Vergnaud. New Techniques for SPHFs and Efficient One-Round PAKE Protocols. In *CRYPTO'13*, volume 8042 of *Lecture Notes in Computer Science*, pages 449–475. Springer-Verlag, 2013. 2, 3, 10
7. O. Blazy, D. Pointcheval, and D. Vergnaud. Round-Optimal privacy-preserving protocols with smooth projective hash functions. In *TCC'12*, volume 7194 of *Lecture Notes in Computer Science*, pages 94–111. Springer-Verlag, 2012. 1
8. J. Brainard and A. Juels. A new two-server approach for authentication with short secrets. In *USENIX'03*, volume 12 of *SSYM'03*, pages 14–14. USENIX Association, 2003. 11
9. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *CRYPOT'98*, volume 1462 of *Lecture Notes in Computer Science*, pages 13–25. Springer-Verlag, 1998. 1
10. R. Cramer and V. Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *EUROCRYPT'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer-Verlag, 2002. 1, 2
11. R. Gennaro and Y. Lindell. A Framework for Password-Based Authenticated Key Exchange. *ACM Trans. Inf. Syst. Secur.*, 9(2):181–234, may 2006. 1, 2, 3, 4, 5, 12, 14
12. C. Gentry, P. D. MacKenzie, and Z. Ramzan. A Method for Making Password-Based Key Exchange Resilient to Server Compromise. In *CRYPTO'06*, volume 4117 of *Lecture Notes in Computer Science*, pages 142–159. Springer-Verlag, 2006. 10
13. J. Katz, P. MacKenzie, G. Taban, and V. Gligor. Two-server password-only authenticated key exchange. In *ACNS'05*, volume 3531 of *Lecture Notes in Computer Science*, pages 1–16. Springer-Verlag, 2005. 1, 10, 11, 12, 13
14. J. Katz, R. Ostrovsky, and M. Yung. Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords. In *EUROCRYPT'01*, volume 2045 of *Lecture Notes in Computer Science*, pages 475–494. Springer-Verlag, 2001. 1, 11, 12, 14
15. J. Katz and V. Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC'11*, volume 6597 of *Lecture Notes in Computer Science*, pages 293–310. Springer-Verlag, 2011. 1, 2, 3, 4, 6, 11
16. P. MacKenzie, T. Shrimpton, and M. Jakobsson. Threshold password-authenticated key exchange. In *CRYPTO'02*, volume 2442 of *Lecture Notes in Computer Science*, pages 1–41. Springer-Verlag, 2002. 11
17. M. D. Raimondo and R. Gennaro. Provably secure threshold password-authenticated key exchange. In *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 507–523. Springer-Verlag, 2003. 10
18. M. Szydlo and B. Kaliski. Proofs for Two-Server Password Authentication. In *CT-RSA'05*, volume 3376 of *Lecture Notes in Computer Science*, pages 227–244. Springer-Verlag, 2005. 11
19. T. Wu. RFC 2945 - The SRP Authentication and Key Exchange System, sep 2000. 10

A SPHF Definitions

A.1 GL-SPHF [11]

In [11] Gennaro and Lindell formally introduced the first use of SPHF in the PAKE setting, denoted by GL-SPHF here, by "explaining" the KOY protocol from [14]. To describe GL-SPHF on labelled Cramer-Shoup ciphertexts in the framework from [5] we define the following variables: Let $\Gamma(C) = \begin{pmatrix} g_1 & g_2 & h & c \\ 1 & 1 & 1 & d^{\xi} \end{pmatrix}^T \in \mathbb{G}^{4\times 2}$, $\lambda = r \in \mathbb{Z}_p$ and $\Theta_{\texttt{aux}}(C) = (u_1, u_2, e/m, v) \in \mathbb{G}^{1\times 4}$. GL-SPHF then is defined as follows:

- $\mathtt{k_h} \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\texttt{aux}}) : \mathtt{k_h} = (\eta, \theta, \mu, \nu) \in_R \mathbb{Z}_p^{1\times 4}$
- $\mathtt{k_p} \leftarrow \mathtt{KGen_P}(\mathtt{k_h}, L_{\texttt{aux}}, C)$:

$$\mathtt{k_p} = \Gamma \odot \mathtt{k_h} = \begin{pmatrix} g_1 & g_2 & h & c \\ 1 & 1 & 1 & d^{\xi} \end{pmatrix}^T \odot (\eta, \theta, \mu, \nu) = g_1^{\eta} g_2^{\theta} h^{\mu} c^{\nu} d^{\xi\nu} \in \mathbb{G}$$

- $h \leftarrow \mathtt{Hash}(\mathtt{k_h}, L_{\texttt{aux}}, C)$:

$$h = \Theta_{\texttt{aux}}(C) \odot \mathtt{k_h} = (u_1, u_2, e/m, v) \odot (\eta, \theta, \mu, \nu) = u_1^{\eta} u_2^{\theta} (e/m)^{\mu} v^{\nu} \in \mathbb{G}$$

- $h \leftarrow \mathtt{PHash}(\mathtt{k_p}, L_{\texttt{aux}}, C, r)$:

$$h = \lambda \odot \mathtt{k_p} = r \odot g_1^{\eta} g_2^{\theta} h^{\mu} c^{\nu} d^{\xi\nu} = (g_1^{\eta} g_2^{\theta} h^{\mu} c^{\nu} d^{\xi\nu})^r \in \mathbb{G}$$

A.2 Cramer-Shoup SPHF^x

The Cramer-Shoup SPHF^x is fully defined as follows (please see Section 3.3 for specifications of $\Theta^x_{\texttt{aux}}, \Theta^0_{\texttt{aux}}$ and Γ):

- $\mathtt{k_h}{}_i \overset{R}{\leftarrow} \mathtt{KGen_H}(L_{\texttt{aux}}) : \mathtt{k_h}{}_i = (\eta_1, \eta_2, \theta, \mu, \nu) \in_R \mathbb{Z}_p^{1\times 5}$.
- $\mathtt{k_p}{}_i \leftarrow \mathtt{KGen_P}(\mathtt{k_h}{}_i, L_{\texttt{aux}})$:

$$\mathtt{k_p}{}_i = \Gamma \odot \mathtt{k_h}{}_i = \begin{pmatrix} g_1 & 1 & g_2 & h & c \\ 1 & g_1 & 1 & 1 & d \end{pmatrix} \odot (\eta_{1,i}, \eta_{2,i}, \theta_i, \mu_i, \nu_i) = \begin{pmatrix} g_1^{\eta_{1,i}} g_2^{\theta_i} h^{\mu_i} c^{\nu_i} \\ g_1^{\eta_{2,i}} d^{\nu_i} \end{pmatrix} \in \mathbb{G}^{2\times 1}$$

- $h_x \leftarrow \mathtt{Hash}_x(\mathtt{k_h}{}_0, L_{\texttt{aux}}, C_1, \ldots, C_x)$:

$$\begin{aligned} h_x &= \Theta^x_{\texttt{aux}}(C_1, \ldots, C_x) \odot \mathtt{k_h}{}_0 \\ &= (\prod_{i=1}^{x} u_{1,i}, \prod_{i=1}^{x} u_{1,i}^{\xi_i}, \prod_{i=1}^{x} u_{2,i}, \prod_{i=1}^{x} e_i/m, \prod_{i=1}^{x} v_i) \odot (\eta_{1,0}, \eta_{2,0}, \theta_0, \mu_0, \nu_0) \\ &= (\prod_{i=1}^{x} u_{1,i})^{\eta_{1,0}} \cdot (\prod_{i=1}^{x} u_{1,i}^{\xi_i})^{\eta_{2,0}} \cdot (\prod_{i=1}^{x} u_{2,i})^{\theta_0} \cdot (\prod_{i=1}^{x} e_i/m)^{\mu_0} \cdot (\prod_{i=1}^{x} v_i)^{\nu_0} \in \mathbb{G} \end{aligned}$$

- $h_x \leftarrow \mathtt{PHash}_x(\mathtt{k_p}{}_0, L_{\texttt{aux}}, C_1, \ldots, C_x, r_1, \ldots, r_x)$: with $\Omega(r_i, C_i) = (r_i, r_i\xi_i)$

$$\begin{aligned} h_x &= \prod_{i=1}^{x} (\lambda^i \odot \mathtt{k_p}{}_0) = \prod_{i=1}^{x} \left((r_i, r_i\xi_i) \odot \begin{pmatrix} g_1^{\eta_{1,0}} g_2^{\theta_0} h^{\mu_0} c^{\nu_0} \\ g_1^{\eta_{2,0}} d^{\nu_0} \end{pmatrix} \right) \\ &= \prod_{i=1}^{x} [(g_1^{\eta_{1,0}} g_2^{\theta_0} h^{\mu_0} c^{\nu_0})^{r_i} (g_1^{\eta_{2,0}} d^{\nu_0})^{r_i\xi_i}] \in \mathbb{G} \end{aligned}$$

- $h_0 \leftarrow \mathtt{Hash}_0(\mathtt{k_h}{}_1, \ldots, \mathtt{k_h}{}_x, L_{\texttt{aux}}, C_0)$:

$$\begin{aligned} h_0 &= \prod_{i=1}^{x} (\Theta^0_{\texttt{aux}}(C_0) \odot \mathtt{k_h}{}_i) = \prod_{i=1}^{x} [(u_1, u_1^{\xi}, u_2, e/m, v) \odot (\eta_{1,i}, \eta_{2,i}, \theta_i, \mu_i, \nu_i)] \\ &= \prod_{i=1}^{x} (u_1^{\eta_{1,i}} \cdot u_1^{\xi\eta_{2,i}} \cdot u_2^{\theta_i} \cdot (e/m)^{\mu_i} \cdot v^{\nu_i}) \end{aligned}$$

– $h_0 \leftarrow \texttt{PHash}_0(\texttt{k}_{\texttt{p}_1}, \ldots, \texttt{k}_{\texttt{p}_x}, L_{\texttt{aux}}, C_0, r_0)$: with $\lambda^0 = \Omega(r_0, C_0)$

$$h_0 = \prod_{i=1}^{x} (\lambda^0 \odot \texttt{k}_{\texttt{p}_i}) = \prod_{i=1}^{x} \left((r_0, r_0\xi_0) \odot \begin{pmatrix} g_1^{\eta_{1,i}} g_2^{\theta_i} h^{\mu_i} c^{\nu_i} \\ g_1^{\eta_{2,i}} d^{\nu_i} \end{pmatrix} \right)$$
$$= \prod_{i=1}^{x} [(g_1^{\eta_{1,i}} g_2^{\theta,i} h^{\mu_i} c^{\nu_i})^{r_0} (g_1^{\eta_{2,i}} d^{\nu_i})^{r_0\xi_0}] \in \mathbb{G}$$

A.3 ElGamal SPHF^x

We use m for the encrypted message, that is part of $\texttt{aux}'$ and $\texttt{pk}$ for the ElGamal public key $h = g^z$ from the $\texttt{crs}$. The ciphertexts are created as $C_i = (u, e) \leftarrow \texttt{Enc}^{\text{EG}}_{\texttt{pk}}(m_i; r_i)$ for all $i = 1, \ldots, x$ with $m_i = h(m)[i]$ and $C_0 = (u, e) \leftarrow \texttt{Enc}^{\text{EG}}_{\texttt{pk}}(m; r_0)$. The decryption follows the ElGamal decryption such that $\texttt{Dec}'_\pi = \texttt{Dec}^{\text{EG}}_z$. The combining function g uses the homomorphic property of u and e such that $g(C_1, \ldots, C_x) = (\prod_{i=1}^x u_i, \prod_{i=1}^x e_i)$. To use the SPHF framework we also need the following variables and functions:

$$\Gamma(C) = (g, h)^T \in \mathbb{G}^{2\times 1}, \quad \lambda = r \in \mathbb{Z}_p, \quad \Theta^0_{\texttt{aux}}(C) = (u, e/m) \in \mathbb{G}^{1\times 2}$$
$$\Theta^x_{\texttt{aux}}(C_1, \ldots, C_x) = (\prod_{i=1}^{x} u_i, \prod_{i=1}^{x} e_i/m) \in \mathbb{G}^{1\times 2}$$

Using them in the SPHF^x Definition 4 yields the following ElGamal SPHF^x:

– $h_0 \leftarrow \texttt{Hash}_0(\texttt{k}_{\texttt{h}1}, \ldots, \texttt{k}_{\texttt{h}x}, L_{\texttt{aux}}, C_0)$:

$$h_0 = \prod_{i=1}^{x} \Theta^0_{\texttt{aux}}(C) \odot \texttt{k}_{\texttt{h}i} = \prod_{i=1}^{x} [(u_0, e_0/m) \odot (\eta, \theta)] = \prod_{i=1}^{x} [u_0^{\eta_i} (e_0/m)^{\theta_i}] \in \mathbb{G}$$

– $h_x \leftarrow \texttt{PHash}_x(\texttt{k}_{\texttt{p}_0}, L_{\texttt{aux}}, C_1, \ldots, C_x, r_1, \ldots, r_x)$:

$$h_x = \prod_{i=1}^{x} (\lambda^i \odot \texttt{k}_{\texttt{p}_0}) = \prod_{i=1}^{x} (r_i \odot g^{\eta_0} h^{\theta_0}) = \prod_{i=1}^{x} (g^{\eta_0} h^{\theta_0})^{r_i} \in \mathbb{G}$$

B SPHF Technicalities

B.1 Smoothness of Cramer-Shoup SPHF^x

We want to discuss the statistical smoothness of SPHF^x from Theorem 1 in this section. While the intuition and actual proof has been given in Section 3, we want to formulate what actually happens there. Therefore, we use the instantiation of Cramer-Shoup SPHF^x and limit $x = 2$. Recall that we thus want to show that $(\texttt{k}_{\texttt{p}_0}, \texttt{k}_{\texttt{p}_1}, \texttt{k}_{\texttt{p}_2}, h_0, h_x)$ is uniformly distributed in $\mathbb{G}^{k+2}$ for all $C \notin L_{\texttt{aux}}$. Actually, we do not have to bother with the projection keys $\texttt{k}_{\texttt{p}_0}, \texttt{k}_{\texttt{p}_1}, \texttt{k}_{\texttt{p}_2}$, as they are each uniformly at random in $\mathbb{G}^k$ anyway, given the randomness of all $\texttt{k}_{\texttt{h}i}$. What we want to show is that given $\texttt{k}_{\texttt{p}_0}, \texttt{k}_{\texttt{p}_1}, \texttt{k}_{\texttt{p}_2}$, the hash values h_0 and h_x are uniformly distributed in $\mathbb{G}$. More precisely, we show that for all $C = (C_0, C_1, C_2) \notin L_{\texttt{aux}}$ the projection keys $\texttt{k}_{\texttt{p}_0}, \texttt{k}_{\texttt{p}_1}, \texttt{k}_{\texttt{p}_2}$ are defined by functions that are linearly independent from the functions used in $\texttt{Hash}_0$ and $\texttt{Hash}_x$, such that the resulting hash values $h_0 \leftarrow \texttt{Hash}_0$ and $h_x \leftarrow \texttt{Hash}_x$ are uniformly distributed in $\mathbb{G}$. Computing the discrete logarithm in base g_1 of h_x, h_0 and the projection keys $\texttt{k}_{\texttt{p}_1}, \texttt{k}_{\texttt{p}_1}$ and $\texttt{k}_{\texttt{p}_2}$ with $m = g^{\text{pw}}$ and $m' = g^{\text{pw}'}$ such that $\texttt{Enc}^{\text{CS}}_{\texttt{pk}}(m') \notin L_{\texttt{aux}}$ we get the following equations:

$$\begin{aligned} \log_{g_1}(h_0) =& \log_{g_1}(\texttt{k}_{\texttt{p}_1}[0]) \cdot r_0 + \log_{g_1}(\texttt{k}_{\texttt{p}_1}[1]) \cdot \xi_0 r_0 \\ &+ \log_{g_1}(\texttt{k}_{\texttt{p}_2}[0]) \cdot r_0 + \log_{g_2}(\texttt{k}_{\texttt{p}_2}[1]) \cdot \xi_0 r_0 + \log_{g_1}(m'/m) \cdot (\nu_1 + \nu_2) \\ =& r_0(\eta_{1,1} + \eta_{1,2}) + \xi_0 r_0(\eta_{2,1} + \eta_{2,2}) + \log_{g_1}(g_2) \cdot r_0(\theta_1 + \theta_2) + z \cdot r_0(\nu_1 + \nu_2) \\ &+ \log_{g_1}(c) \cdot r_0(\nu_1 + \nu_2) + \log_{g_1}(d) \cdot \xi_0 r_0(\nu_1 + \nu_2) + \log_{g_1}(m'/m) \cdot (\nu_1 + \nu_2) \end{aligned} \quad (1)$$

$$\begin{aligned}\log_{g_1}(h_x) =& \log_{g_1}(\mathtt{k_{p_0}}[0]) \cdot (r_1 + r_2) + \log_{g_1}(\mathtt{k_{p_0}}[1]) \cdot (\xi_1 r_1 + \xi_2 r_2) + \log_{g_1}(m'/m) \cdot \nu_0 \\ =& (r_1 + r_2)\eta_{1,0} + (\xi_1 r_1 + \xi_2 r_2)\eta_{2,0} + \log_{g_1}(g_2) \cdot (r_1 + r_2)\theta_0 + z \cdot (r_1 + r_2)\nu_0 \\ &+ \log_{g_1}(c) \cdot (r_1 + r_2)\nu_0 + \log_{g_1}(d) \cdot (\xi_1 r_1 + \xi_2 r_2)\nu_0 + \log_{g_1}(m'/m) \cdot \nu_0\end{aligned} \tag{2}$$

$$\log_{g_1}(\mathtt{k_{p_0}}[0]) = \eta_{1,0} + \log_{g_1}(g_2) \cdot \theta_0 + \log_{g_1}(h) \cdot \mu_0 + \log_{g_1}(v) \cdot \nu_0 \tag{3}$$

$$\log_{g_1}(\mathtt{k_{p_0}}[1]) = \eta_{2,0} + \log_{g_1}(d) \cdot \nu_0 \tag{4}$$

$$\log_{g_1}(\mathtt{k_{p_1}}[0]) = \eta_{1,1} + \log_{g_1}(g_2) \cdot \theta_1 + z \cdot \mu_1 + \log_{g_1}(v) \cdot \nu_1 \tag{5}$$

$$\log_{g_1}(\mathtt{k_{p_1}}[1]) = \eta_{2,1} + \log_{g_1}(d) \cdot \nu_1 \tag{6}$$

$$\log_{g_1}(\mathtt{k_{p_2}}[0]) = \eta_{1,2} + \log_{g_1}(g_2) \cdot \theta_2 + z \cdot \mu_2 + \log_{g_1}(v) \cdot \nu_2 \tag{7}$$

$$\log_{g_1}(\mathtt{k_{p_2}}[1]) = \eta_{2,2} + \log_{g_1}(d) \cdot \nu_2. \tag{8}$$

Since $C \notin L_{\mathtt{aux}}$ we know that $m \neq m'$ and thus $m'/m \neq 1$. Therefore, the probability for $g_0 = \log_{g_1}(m'/m) \cdot (\nu_1 + \nu_2)$ and $g_x = \log_{g_1}(m'/m) \cdot \nu_0$ for any $g_0, g_x \in \mathbb{G}$ is $1/|\mathbb{G}|$ even knowing the projection keys $\mathtt{k_{p_0}}, \mathtt{k_{p_1}}, \mathtt{k_{p_2}}$. Note that these equations for g_0 and g_x are linearly independent from Equations 3 - 8 such that every element from $\mathbb{G}$ is equally likely to be the result. Equations 1 and 2 are fully determined by public information C_0, C_1, C_2 and $\mathtt{k_{p_0}}, \mathtt{k_{p_1}}, \mathtt{k_{p_2}}$ such that their result is uniformly distributed in $\mathbb{G}$ given the randomness of g_0 and g_x.

C Notation

Using common matrix and vector operations on $\odot$ can be used as follows:

$$b \odot s = \begin{pmatrix} b_{11} & \cdots & b_{1y} \\ \vdots & & \vdots \\ b_{x1} & \cdots & b_{xy} \end{pmatrix} \odot (s_{11}, \ldots, s_{1y}) = (\prod_{i=1}^{y} b_{1i}^{s_{1i}}, \ldots, \prod_{i=1}^{y} b_{xi}^{s_{xi}})^T \in \mathbb{G}^{x \times 1}$$

$$s \odot c = (s_{11}, \ldots, s_{1y}) \odot (c_{11}, \ldots, c_{y1})^T = \prod_{i=1}^{y} c_{i1}^{s_{1i}} \in \mathbb{G}$$