A Framework for UC-Secure Commitments from Publicly Computable Smooth Projective Hashing

Behzad Abdolmaleki[1], Hamidreza Khoshakhlagh[2,⋆], and Daniel Slamanig[3]

[1] University of Tartu, Estonia
behzad.abdolmaleki@ut.ee
[2] Aarhus University, Denmark
hamidreza@cs.au.dk
[3] AIT Austrian Institute of Technology, Vienna, Austria
daniel.slamanig@ait.ac.at

Abstract. Hash proof systems or smooth projective hash functions (SPHFs) have been proposed by Cramer and Shoup (Eurocrypt'02) and can be seen as special type of zero-knowledge proof system for a language. While initially used to build efficient chosen-ciphertext secure public-key encryption, they found numerous applications in several other contexts. In this paper, we revisit the notion of SPHFs and introduce a new feature (a third mode of hashing) that allows to compute the hash value of an SPHF without having access to neither the witness nor the hashing key, but some additional auxiliary information. We call this new type publicly computable SPHFs (PC-SPHFs) and present a formal framework along with concrete instantiations from a large class of SPHFs.
We then show that this new tool generically leads to commitment schemes that are secure against adaptive adversaries, assuming erasures in the Universal Composability (UC) framework, yielding the first UC secure commitments build from a single SPHF instance. Instantiating our PC-SPHF with an SPHF for labeled Cramer-Shoup encryption gives the currently most efficient non-interactive UC-secure commitment. Finally, we also discuss additional applications to information retrieval based on anonymous credentials being UC secure against adaptive adversaries.

1 Introduction

Hash proof systems or smooth projective hash functions (SPHFs) were introduced by Cramer and Shoup [CS02] and can be considered as implicit designated-verifier proofs of membership [ACP09,BPV12]. Similarly to zero-knowledge proofs, SPHFs are defined for a NP language $\mathcal{L}$ and one considers membership of words $\mathtt{x} \in \mathcal{L}$. In SPHFs, a verifier can generate a secret hashing key hk and for any word $\mathtt{x}$ she can compute a hash value H by using the hashing key hk and $\mathtt{x}$. In addition, the verifier can derive a projection key hp from the hashing key hk and send it to the prover. By knowing a witness $\mathtt{w}$ for membership of $\mathtt{x} \in \mathcal{L}$ and having the projection key hp, the prover is able to efficiently compute the

[⋆] Majority of this work was done while working at the University of Tartu.

projected hash pH for the word $\mathtt{x}$ such that it equals the hash H computed by the verifier. The smoothness property says that if $\mathtt{x} \notin \mathcal{L}$ one cannot guess the hash value H by knowing hp, or in other words, H looks completely random.

One of the very fundamental tools in cryptographic protocols are commitment schemes. They allow a committer C to pass an analogue of a sealed envelope of his message m to a receiver R. When the committer C then later reveals m with some additional opening information, R can verify whether the envelop contains m. It should be guaranteed that C cannot change the committed message m to some $m' \neq m$ later (binding property) and that R must not learn any information about the committed message m in the commit phase before the opening (hiding property). Some well known perfectly binding commitment are the Cramer-Shoup (CS) [CS02] and ElGamal [ElG84] encryption schemes, or Pedersen commitments [Ped92] for the case of perfectly hiding commitments.

To be suitable for the use within the universal composability (UC) framework [Can01], commitment schemes need to provide strong properties and in particular extractability and equivocability. The first one states that the simulator Sim can recover the committed value m by knowing a trapdoor and the latter means that Sim can open a commitment to any message $m' \neq m$ by means of a trapdoor. Satisfying both properties turns out to be a rather difficult task. In general, constructing efficient equivocable and extractable commitments falls into two categories: the one following the ideas of Canetti and Fischlin [Can01] including [ACP09,BBC$^+$13,ABP17,ABL$^+$19], and the ones using non-interactive zero-knowledge proofs as the opening information as the Fischlin-Libert-Manulis schemes [FLM11] and improvements thereof [JR13]. In this paper, we go into latter direction, but instead of non-interactive zero-knowledge proofs, we use the PC-SPHF which allows us to improve the communication complexity.

Our Contribution. We first introduce an extension of classical SPHFs which we call publicly computable SPHFs (PC-SPHFs) in Section 3. Our focus is on SPHFs for languages of ciphertexts $\mathcal{L}_{\mathsf{aux}}$, parametrized by aux, instantiated in the source groups of a bilinear group, i.e., which are itself pairing-free. This covers many schemes such as (linear) ElGamal or (linear) CS. A PC-SPHF is such an SPHF with the following additional property: there is a third mode of hashing which allows to compute the hash value in the target group $\mathbb{G}_T$ of a bilinear group when neither having access to the hashing key hk nor the witness $\mathtt{w}$. This is achieved by adding some representations of the hashing key hk in the projection key hp such that by using aux, hp, and some public values ($\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$), one can compute the hash value in $\mathbb{G}_T$.

We then in Section 4 show how one can use PC-SPHFs built from any suitable SPHF for a labeled IND-CCA encryption scheme to construct a generic UC-secure commitment scheme. Following this approach, we construct the most efficient non-interactive UC-secure commitment by using the labeled CS encryption scheme ($\mathsf{PC\text{-}SPHF_{CS}}$). We compare the efficiency of $\mathsf{PC\text{-}SPHF_{CS}}$ with existing non-interactive UC-secure commitments in Table 1[4] and, as we discuss

[4] We note that we follow existing literature and thus focus on the size of commitments and openings and exclude the message(s) in the opening information.

in Section 4.2, this gives us an improvement of around 30% over the UC-secure commitments in [ABP17], which to the best of our knowledge represent the most efficient UC-secure commitments to date. Compared to the most efficient UC-secure commitments in bilinear groups, we obtain an improvement in the opening of a factor 4.

Scheme	|Commitment|	|Opening|	Assumption
[CF01]	$9 \times \mathbb{G}$	$2 \times \mathbb{Z}_p$	Plain DDH
[FLM11],1	$5 \times \mathbb{G}_1$	$16 \times \mathbb{G}_1$	DLIN
[FLM11],2	$37 \times \mathbb{G}_1$	$3 \times \mathbb{G}_1$	DLIN
[JR13]	$4 \times \mathbb{G}_1$	$3 \times \mathbb{G}_1 + 2 \times \mathbb{G}_2$	SXDH
[JR13]	$4 \times \mathbb{G}_1$	$4 \times \mathbb{G}_1$	DLIN
[ABB+13]	$8 \times \mathbb{G}_1 + \mathbb{G}_2$	$\mathbb{Z}_p$	SXDH
[ABP17]	$7 \times \mathbb{G}$	$2 \times \mathbb{Z}_p$	Plain DDH
$\mathsf{PC\text{-}SPHF}_{\mathsf{CS}}$	$4 \times \mathbb{G}_1$	$\mathbb{G}_1$	XDH

Table 1. Comparison with some existing non-interactive UC-secure commitments with a single global CRS when committing to a single message.

Finally, in Section 5 we show how PC-SPHFs help to improve the efficiency of information retrieval based on anonymous credentials (as proposed in [BC16]), which is UC secure against adaptive adversaries. In a nutshell, such protocols use anonymous credentials to securely retrieve a message without revealing the identity of the receiver to the sender.

2 Preliminaries

Let PPT denote probabilistic polynomial-time. Let $\lambda \in \mathbb{N}$ be the security parameter. All adversaries will be stateful. By $y \leftarrow \mathcal{A}(\mathtt{x}; r)$ we denote the fact that $\mathcal{A}$, given an input $\mathtt{x}$ and randomness r, outputs y. By $x \leftarrow_{\$} \mathcal{D}$ we denote that x is sampled according to distribution $\mathcal{D}$ or uniformly random if $\mathcal{D}$ is a set. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function. A bilinear group generator $\mathsf{Pgen}(1^\lambda)$ returns $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \bar{e})$, where $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three cyclic groups of prime order p, and $\bar{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a non-degenerate efficiently computable bilinear pairing. We use the implicit bracket notation of [EHK+13], that is, we write $[a]_\iota$ to denote ag_ι where g_ι is a fixed generator of $\mathbb{G}_\iota$. We denote $\bar{e}([a]_1, [b]_2)$ as $[a]_1 \bullet [b]_2$. Thus, $[a]_1 \bullet [b]_2 = [ab]_T$. We denote $s[a]_\iota = [sa]_\iota$ for $s \in \mathbb{Z}_p$ and $\iota \in \{1, 2, T\}$. We freely use the bracket notation together with matrix notation, for example, if $\boldsymbol{AB} = \boldsymbol{C}$ then $[\boldsymbol{A}]_1 \bullet [\boldsymbol{B}]_2 = [\boldsymbol{C}]_T$.

Labeled Public-Key Encryption. Labeled encryption is a variant of the public-key encryption notion where encryption and decryption take an additional label. Decryption should only work if the label used for decryption is identical to the one used when producing the ciphertext. More formally:

Definition 1. *A labeled (tagged) public-key encryption scheme* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$*, is defined by three algorithms:*

- $\mathsf{KGen}(1^\lambda)$ *given a security parameter* λ*, generates a public key* pk *and a secret key* sk*.*
- $\mathsf{Enc}^{\mathtt{t}}_{\mathsf{pk}}(M)$ *given the public key* pk*, a label* $\mathtt{t}$ *and a message* M*, outputs a ciphertext* $\mathbf{c}$*.*
- $\mathsf{Dec}^{\mathtt{t}}_{\mathsf{sk}}(\mathbf{c})$ *given a label* $\mathtt{t}$*, the secret key* sk *and a ciphertext* $\mathbf{c}$*, with* $\mathbf{c} = \mathsf{Enc}^{\mathtt{t}}_{\mathsf{pk}}(M)$*, then outputs* M*.*

For correctness it is required that for all $(\mathsf{pk}, \mathsf{sk}) \in \mathsf{KGen}(1^\lambda)$, all labels $\mathtt{t}$ and all messages M, $\mathsf{Dec}^{\mathtt{t}}_{\mathsf{sk}}(\mathsf{Enc}^{\mathtt{t}}_{\mathsf{pk}}(M)) = M$. Henceforth, we use public-key encryption schemes that provide indistinguishability under chosen plaintext and adaptive chosen ciphertexts attacks, i.e., provide IND-CPA or IND-CCA security respectively.

Decisional Diffie-Hellman (DDH) Assumption. Let $\iota \in \{1, 2, T\}$. DDH holds relative to Pgen, for all λ, all PPT adversaries $\mathcal{A}$, $|\mathsf{Exp}^{\mathsf{DDH}}_{\mathcal{A}}(\mathsf{pars}) - 1/2| \approx_\lambda 0$, where $\mathsf{Exp}^{\mathsf{DDH}}_{\mathcal{A}}(\mathsf{pars}) :=$

$$\Pr\left[\begin{matrix}\mathsf{pars} \leftarrow_\$ \mathsf{Pgen}(1^\lambda); u, v, w \leftarrow_\$ \mathbb{Z}_p; b \leftarrow_\$ \{0,1\}; \\ b^* \leftarrow \mathcal{A}(\mathsf{pars}, [u]_\iota, [v]_\iota, [b \cdot uv + (1-b)w]_\iota)\end{matrix} : b = b^*\right].$$

Smooth Projective Hash Functions. Smooth projective hash functions (SPHFs) [CS02] are families of pairs of functions $(\mathsf{hash}, \mathsf{projhash})$ defined on a language $\mathcal{L}$. They are indexed by a pair of associated keys $(\mathsf{hk}, \mathsf{hp})$, where the hashing key hk may be viewed as the private key and the projection key hp as the public key. On a word $\mathtt{x} \in \mathcal{L}$, both functions need to yield the same result, that is, $\mathsf{hash}(\mathsf{hk}, \mathcal{L}, \mathtt{x}) = \mathsf{projhash}(\mathsf{hp}, \mathcal{L}, \mathtt{x}, \mathtt{w})$, where the latter evaluation additionally requires a witness $\mathtt{w}$ that $\mathtt{x} \in \mathcal{L}$. Thus, they can be seen as a tool for implicit designated-verifier proofs of membership [ACP09]. Formally SPHFs are defined as follows (cf. [BBC+13]).

Definition 2. *A SPHF for a language* $\mathcal{L}$ *is a tuple of PPT algorithms* $(\mathsf{Pgen}, \mathsf{hashkg}, \mathsf{projkg}, \mathsf{hash}, \mathsf{projhash})$*, which are defined as follows:*

$\mathsf{Pgen}(1^\lambda)$**:** *Takes a security parameter* λ *and generates the global parameters* pars *(we assume that all algorithms have access to* pars*).*
$\mathsf{hashkg}(\mathcal{L})$**:** *Takes a language* $\mathcal{L}$ *and outputs a hashing key* hk *for* $\mathcal{L}$*.*
$\mathsf{projkg}(\mathsf{hk}, \mathcal{L}, \mathtt{x})$**:** *Takes a hashing key* hk*,* $\mathcal{L}$*, and a word* $\mathtt{x}$ *and outputs a projection key* hp*, possibly depending on* $\mathtt{x}$*.*
$\mathsf{hash}(\mathsf{hk}, \mathcal{L}, \mathtt{x})$**:** *Takes a hashing key* hk*,* $\mathcal{L}$*, and a word* $\mathtt{x}$ *and outputs a hash* H*.*
$\mathsf{projhash}(\mathsf{hp}, \mathcal{L}, \mathtt{x}, \mathtt{w})$**:** *Takes a projection key* hp*,* $\mathcal{L}$*, a word* $\mathtt{x}$*, and a witness* $\mathtt{w}$ *for* $\mathtt{x} \in \mathcal{L}$ *and outputs a hash* pH*.*

A SPHF needs to satisfy the following properties:

Correctness. It is required that $\mathsf{hash}(\mathsf{hk}, \mathcal{L}, \mathtt{x}) = \mathsf{projhash}(\mathsf{hp}, \mathcal{L}, \mathtt{x}, \mathtt{w})$ for all $\mathtt{x} \in \mathcal{L}$ and their corresponding witnesses $\mathtt{w}$.

Smoothness. It is required that for any pars and any word $\mathtt{x} \notin \mathcal{L}$, the following distributions are statistically indistinguishable:

$$\{(\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{L}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathcal{L}, \mathtt{x}), \mathsf{H} \leftarrow \mathsf{hash}(\mathsf{hk}, \mathcal{L}, \mathtt{x})\}$$
$$\{(\mathsf{hp}, \mathsf{H}) : \mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{L}), \mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathcal{L}, \mathtt{x}), \mathsf{H} \leftarrow_{\$} \Omega\} \quad .$$

where Ω is the set of hash values.

Depending on the definition of smoothness, there are three types of SPHFs (cf. [BBC+13]):

GL-SPHF. The projection key hp can depend on word $\mathtt{x}$ and so the smoothness is correctly defined only if $\mathtt{x}$ is chosen before having seen hp.

KV-SPHF. hp does not depend on word $\mathtt{x}$ and the smoothness holds even if $\mathtt{x}$ is chosen after having seen hp.

CS-SPHF. hp does not depend on word $\mathtt{x}$ but the smoothness holds only if $\mathtt{x}$ is chosen before having seen hp.

Trapdoor Smooth Projective Hash Functions. Benhamouda et. al [BBC+13] proposed an extension of a classical SPHF, called TSPHF. Their construction has an additional algorithm tsetup, which takes as input the CRS $\mathtt{crs}$ (output by Pgen) and outputs an additional CRS $\mathtt{crs}_\tau$ with trapdoor τ, which can be used to compute the hash value of words $\mathtt{x}$ knowing hp and the trapdoor τ. We refer the reader to [BBC+13] for a rigorous formal definition of TSPHFs, and only briefly discuss their computational smoothness property.

Smoothness is the same as for SPHFs, except that after calling Pgen the trapdoor τ of the $\mathtt{crs}_\tau$ is dropped, but $\mathtt{crs}_\tau$ is forwarded to the adversary (together with $\mathtt{crs}_\mathcal{L}$ and even its trapdoor $\mathtt{tds}$). Notice that since hp now needs to contain enough information to compute the hash value of any word $\mathtt{x}$, the smoothness property of TSPHFs is no longer statistical but computational.

2.1 SPHFs on Languages of Ciphertexts

In this paper we mainly focus on SPHFs for languages of ciphertexts, whose corresponding plaintexts verify some relations. The language parameters parse in two parts ($\mathtt{crs}_\mathcal{L}, \mathsf{aux}$): the public part $\mathtt{crs}_\mathcal{L}$, known in advance, and the private part aux, possibly chosen later. More concretely, $\mathtt{crs}_\mathcal{L}$ represents the public values: it will define the (labeled) encryption scheme (and will thus contain the global parameters and the public key of the (labeled) encryption scheme) with the global format of both the tuple to be encrypted and the relations it should satisfy, and possibly additional public coefficients. While aux represents the private values: it will specify the relations, with more coefficients or constants that will remain private, i.e,. the message encrypted in the ciphertext in such languages. We will henceforth denote such languages by $\mathcal{L}_\mathsf{aux}$. Since in this paper we mostly focus on constructing a UC-secure commitment scheme where the $\mathtt{crs}$ is independent of the ciphertext (the word of the language), we focus on KV-SPHFs as used in [KV11,BBC+13,BBC+15].

Language Representation. Similar to [BBC+13], for a language $\mathcal{L}_{\mathsf{aux}}$, we assume there exist two positive integers k and n, a function $\Gamma : S \to \mathbb{G}^{k \times n}$, and a family of functions $\Theta_{\mathsf{aux}} : S \to \mathbb{G}^{1 \times n}$, such that $\mathtt{x} \in S$ ($\mathtt{x} \in \mathcal{L}_{\mathsf{aux}}$) iff $\exists \boldsymbol{\lambda} \in \mathbb{Z}_p^{1 \times k}$ such that $\Theta_{\mathsf{aux}}(\mathtt{x}) = \boldsymbol{\lambda}\Gamma(\mathtt{x})$. In other words, we assume that $\mathtt{x} \in \mathcal{L}_{\mathsf{aux}}$, if and only if, $\Theta_{\mathsf{aux}}(\mathtt{x})$ is a linear combination of (the exponents in) the rows of some matrix $\Gamma(\mathtt{x})$. For a KV-SPHF, $\boldsymbol{\Gamma}$ is supposed to be a constant function (independent of the word $\mathtt{x}$). Otherwise, one gets a GL-SPHF. We furthermore require that when knowing a witness $\mathtt{w}$ of the membership $\mathtt{x} \in \mathcal{L}_{\mathsf{aux}}$, one can efficiently compute the above linear combination $\boldsymbol{\lambda}$. This may seem a quite strong requirement, but this is actually satisfied by very expressive languages over ciphertexts such as ElGamal, CS and variants.

In the following, we briefly illustrate it on a KV-SPHF for the language of (labeled) CS ciphertexts encrypting a message $[m]_1 \in \mathbb{G}_1$ and $\mathsf{aux} := [m]_1$.

(Labeled) CS Ciphertext Language. The CS IND-CCA2 secure public-key encryption scheme in an abelian cyclic group $\mathbb{G}_1$ of order p is defined as follows: the secret key sk is $(x_1, x_2, y_1, y_2, z) \leftarrow_{\$} \mathbb{Z}_p^5$. Assume $[g_1]_1, [g_2]_1$ are two different independent generators of $\mathbb{G}_1$. Let H be a collision-resistant hash function. The public key is $\mathsf{pk} = ([g_1, g_2, h, c, d]_1, H)$, where $[c]_1 = x_1[g_1]_1 + x_2[g_2]_1$, $[d]_1 = y_1[g_1]_1 + y_2[g_2]_1$, $h = z[g_1]_1$. The encryption of $[m]_1$ with randomness $r \leftarrow_{\$} \mathbb{Z}_p$ is defined as $[\mathbf{c}]_1 = [u_1, u_2, e, v]_1$ where $[u_1]_1 = r[g_1]_1$, $[u_2]_1 = r[g_2]_1$, $[e]_1 = [m]_1 + r[h]_1$, $[v]_1 = r([c]_1 + \xi[d]_1)$, where $\xi = H([u_1]_1, [u_2]_1, [e]_1)$. In case of labeled CS with label t, the hash value is computed as $\xi = H(\mathsf{t}, [u_1]_1, [u_2]_1, [e]_1)$.

Smooth Projective Hash Function for (Labeled) CS Ciphertexts. With the notation introduced earlier, the hashing key is a vector $\mathsf{hk} = \boldsymbol{\alpha} \leftarrow_{\$} \mathbb{Z}_p^n$, while the projection key is, for a word $\mathtt{x}$, $\mathsf{hp} = [\Gamma(\mathtt{x})]_1\boldsymbol{\alpha} \in \mathbb{G}_1^k$ (if $\boldsymbol{\Gamma}$ depends on $\mathtt{x}$, this leads to a GL-SPHF, otherwise, one obtains a KV-SPHF). We have:

$$\mathsf{hash}(\mathsf{hk}, \mathcal{L}_{\mathsf{aux}}, \mathtt{x}) = \Theta_{\mathsf{aux}} \cdot \boldsymbol{\alpha} = \boldsymbol{\lambda} \cdot \mathsf{hp} = \mathsf{projhash}(\mathsf{hp}, \mathcal{L}_{\mathsf{aux}}, \mathtt{x}, \mathtt{w})$$

The parameters $\boldsymbol{\Gamma}$, $\boldsymbol{\lambda}$ and, $\Theta_{[m]_1}$ immediately lead to the KV-SPHF on (labeled) CS, introduced in [BBC+13]: with $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \iota) \leftarrow_{\$} \mathbb{Z}_p^5$, the product with Γ leads to, $\mathsf{hp} = (\mathsf{hp}_1 = \eta_1[g_1]_1 + \theta[g_2]_1 + \mu[h]_1 + \iota[\iota]_1, \mathsf{hp}_2 = \eta_2[g_1]_1 + \iota[d]_1)$ and,

$$\begin{aligned}\mathsf{H} =& \mathsf{hash}(\mathsf{hk}, (\mathsf{pk}, [m]_1), [\mathbf{c}]_1) = (\eta_1 + \xi\eta_2)[u_1]_1 + \theta[u_2]_1 + \mu([e]_1 - [m]_1) + \iota[v]_1 \\ =& r[\mathsf{hp}_1]_1 + r\xi[\mathsf{hp}_2]_1 = \mathsf{projhash}(\mathsf{hp}, (\mathsf{pk}, [m]_1), [\mathbf{c}]_1, r) = \mathsf{pH}.\end{aligned}$$

The analysis showing perfect smoothness can be found in [BBC+15].

3 Publicly Computable SPHFs

In this section we show how to construct Publicly Computable SPHFs (PC-SPHFs) in a bilinear group from SPHFs. Our PC-SPHF framework is similar to the generic framework for SPHFs in [BBC+13] with some slight modifications. Conceptually, the construction of PC-SPHF is inspired by TSPHFs [BBC+15], but with completely different motivations and algorithms. A PC-SPHF is an extension of a classical SPHF and in particular based upon an SPHF which can be constructed in the source groups of a bilinear group, i.e., the SPHF itself is pairing-free. The PC-SPHF builds upon the SPHF and is then instantiated in a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \bar{e})$. The third mode of hashing provides a means to publicly compute a representation of the hash of the underlying SPHF in $\mathbb{G}_T$ without having access to secret information hk and $\mathtt{w}$. Also for PC-SPHFs, the algorithm projkg takes a hashing key hk, a language $\mathcal{L}_{\mathsf{aux}}$, and a word $\mathtt{x}$ and outputs a projection key $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) \in \mathbb{G}_\iota^k \times \mathbb{G}_{3-\iota}^n$, where hp_1 is the projection key of the underlying SPHF and hp_2 is some representation of the hashing key hk. We note the the PC-SPHF is actually defined with respect to a family of languages $\{\mathcal{L}_{\mathsf{aux}}\}_{\mathsf{aux} \in \mathsf{AUX}}$ parametrized by aux, i.e., the message encrypted using the encryption scheme associated to the SPHF, but we will not make this explicit for the sake of readability and will always write $\mathcal{L}_{\mathsf{aux}}$ as well as aux.

Definition of PC-SPHFs. In the following we assume an SPHF on languages of ciphertexts (cf. Section 2.1 for the notation) instantiated in the source groups of a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \bar{e})$ and let $\iota \in \{1, 2\}$ (depending on the concrete SPHF). We recall that the hashing key of the SPHF is a vector $\mathsf{hk} = \boldsymbol{\alpha} \leftarrow_\$ \mathbb{Z}_p^n$, while the projection key is, for a word $\mathtt{x}$, $\mathsf{hp} = [\Gamma(\mathtt{x})]_\iota \boldsymbol{\alpha} \in \mathbb{G}_\iota^k$.

Definition 3. *A PC-SPHF for language $\mathcal{L}_{\mathsf{aux}}$ based upon SPHF is defined by the following algorithms:*

$\mathsf{Pgen}(1^\lambda, \mathcal{L}_{\mathsf{aux}})$**:** *Takes a security parameter λ and language $\mathcal{L}_{\mathsf{aux}}$ and generates the global parameters pars, and the $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$. It outputs $(\mathsf{pars}, \mathsf{aux}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}})$.*

$\mathsf{hashkg}(\mathcal{L}_{\mathsf{aux}})$**:** *Takes a language $\mathcal{L}_{\mathsf{aux}}$ and outputs a hashing key $\mathsf{hk} = \boldsymbol{\alpha} \leftarrow_\$ \mathbb{Z}_p^n$ for the language $\mathcal{L}_{\mathsf{aux}}$ of the underlying SPHF.*

$\mathsf{projkg}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathtt{x})$**:** *Takes a hashing key hk, a CRS $\mathtt{crs}$, and possibly a word $\mathtt{x}$ and outputs a projection key $\mathsf{hp} = (\mathsf{hp}_1, \mathsf{hp}_2) \in \mathbb{G}_\iota^k \times \mathbb{G}_{3-\iota}^n$, possibly depending on $\mathtt{x}$, where hp_1 is the projection key of the underlying SPHF and hp_2 is some representation of hk.*

$\mathsf{hash}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{aux}, \mathtt{x})$**:** *Takes a hashing key hk, a CRS $\mathtt{crs}$, aux, and a word $\mathtt{x}$ and outputs a hash $\mathsf{H} \in \mathbb{G}_\iota$, being the hash of the underlying SPHF.*

$\mathsf{projhash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{aux}, \mathtt{x}, \mathtt{w})$**:** *Takes a projection key hp, a CRS $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$, aux, a word $\mathtt{x}$, and a witness $\mathtt{w}$ for $\mathtt{x} \in \mathcal{L}_{\mathsf{aux}}$ and outputs a hash $\mathsf{pH} \in \mathbb{G}_\iota$, being the projective hash of the underlying SPHF.*

$\mathsf{pchash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{aux}, \mathtt{x})$**:** *Takes a projection key hp, a CRS $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$, aux, and a word $\mathtt{x}$, and outputs a hash $\mathsf{pcH} \in \mathbb{G}_T$.*

A PC-SPHF must satisfy the following properties:

Perfect correctness. For any $(\mathsf{pars}, \mathsf{aux}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}) \leftarrow \mathsf{Pgen}(1^\lambda, \mathcal{L}_{\mathsf{aux}})$ and any word $\mathtt{x} \in \mathcal{L}_{\mathsf{aux}}$ with witness $\mathtt{w}$, for any $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{L}_{\mathsf{aux}})$ and for $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathtt{x})$:

$$\mathsf{pH} \bullet [1]_{3-\iota} = \mathsf{pcH}.$$

The (t, ϵ)-soundness property. For any $(\mathsf{pars}, \mathsf{aux}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}) \leftarrow \mathsf{Pgen}(1^\lambda, \mathcal{L}_{\mathsf{aux}})$, given $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$ and the projection key hp, no adversary running in time at most t can produce a value aux, a word $\mathtt{x}$ and valid witness $\mathtt{w}$ such that $\mathsf{projhash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{aux}, \mathtt{x}, \mathtt{w}) \neq \mathsf{hash}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{aux}, \mathtt{x})$, with probability at least ϵ. Perfect soundness requires that this holds for any t and any $\epsilon > 0$.

Computational Smoothness. The computational smoothness experiment is provided in Fig. 1. For a language $\mathcal{L}_{\mathsf{aux}}$ and adversary $\mathcal{A}$, the advantage is defined as follows:

$$\mathsf{Adv}^{\text{csmooth}}_{\mathcal{L}_{\mathsf{aux}},\mathcal{A}}(\lambda) = |\Pr[\mathsf{Exp}^{csmooth-0}(\mathcal{A},\lambda) = 1] - \Pr[\mathsf{Exp}^{csmooth-1}(\mathcal{A},\lambda) = 1]|.$$

and we require that $\mathsf{Adv}^{\text{csmooth}}_{\mathcal{L}_{\mathsf{aux}},\mathcal{A}}(\lambda) \leq \mathsf{negl}(\lambda)$.

```
$\mathsf{Exp}^{csmooth\text{-}b}(\mathcal{A},\lambda)$

  – $(\mathsf{pars}, \mathsf{aux}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}) \leftarrow \mathsf{Pgen}(1^\lambda, \mathcal{L}_{\mathsf{aux}})$, $\mathtt{x} \leftarrow_{\$} \mathcal{X}_{\mathsf{aux}} \setminus \mathcal{L}_{\mathsf{aux}}$, $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{L}_{\mathsf{aux}})$,
    $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathtt{x})$;
  – If $b = 0$, then $\mathsf{H} \leftarrow \mathsf{hash}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{aux}, \mathtt{x})$, else $\mathsf{H} \leftarrow_{\$} \Omega$;
  – return $\mathcal{A}(\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathtt{x}, \mathsf{hp}, \mathsf{H})$
```

Fig. 1. Experiments $\mathsf{Exp}^{csmooth\text{-}b}$ for computational smoothness.

Security Analysis. The correctness and the perfect soundness are easy to verify from the construction, and so the resulting PC-SPHF is correct and sound. Subsequently, we prove the computational smoothness of the PC-SPHF under the XDH assumption, i.e., the DDH assumption in $\mathbb{G}_\iota$. For the sake of exposition, we assume that the SPHF is instantiated in $\mathbb{G}_1$ below, i.e., $\iota = 1$.

Theorem 1. *Let* XDH *hold, then PC-SPHF is computationally smooth.*

Proof. We first reduce the smoothness to the following computational assumption: for all λ, $\mathsf{pars} \in \mathsf{Pgen}(1^\lambda)$, and PPT adversaries $\mathcal{B}$, $|\mathsf{Exp}^{int}_{\mathcal{B}}(\mathsf{pars}) - 1/2| \approx_\lambda 0$, where $\mathsf{Exp}^{int}_{\mathcal{B}}(\mathsf{pars})$ is depicted in Fig. 2. Let $\mathcal{B}$ be allowed to make only one query to the oracle O to obtain a tuple $(([\boldsymbol{\Gamma}\boldsymbol{\alpha}]_1, [\boldsymbol{\alpha}]_1, [\boldsymbol{x}]_1, [x'_i\alpha_i]_1, [x'_i\alpha_i x_i]_1; [\boldsymbol{\alpha}]_2))_{i\in[n]}$.

Let $\mathcal{A}$ be the adversary against computational smoothness. We now construct the following adversary $\mathcal{B}$ against the *intermediate assumption.*

- $(([\boldsymbol{\Gamma}\boldsymbol{\alpha}]_1, [\boldsymbol{\alpha}]_1, [\boldsymbol{x}]_1, [z_i]_1, [z_i x_i]_1; [\boldsymbol{\alpha}]_2))_{i\in[n]} \leftarrow \mathsf{O}_b([\boldsymbol{\Gamma}]_1)$; where if $b = 0$, $z_i = x'_i\alpha_i$ and $z_i \leftarrow_{\$} \mathbb{Z}_p$ otherwise;
- $\mathsf{hp}_1 \leftarrow [\boldsymbol{\Gamma}\boldsymbol{\alpha}]_1$; $\mathsf{hp}_2 \leftarrow [\boldsymbol{\alpha}]_2$;
- $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}} = [\boldsymbol{\Gamma}]_1$;
- $H \leftarrow \sum_{i=1}^n [z_i]_1$; $\mathtt{x} \leftarrow [\boldsymbol{x}]_1$;
- $b_{\mathcal{A}} \leftarrow \mathcal{A}(\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{hp}, \mathtt{x}, H)$;
- **return** $b' \leftarrow b_{\mathcal{A}}$.

```
$\mathsf{Exp}^{int}_{\mathcal{B}}(\mathsf{pars})$
    $([\boldsymbol{\Gamma}]_1, \boldsymbol{\Gamma}) \leftarrow \mathsf{Pgen}^*(\mathsf{pars}, \mathcal{L}_{\mathsf{aux}})$;
    $b \leftarrow_\$ \{0,1\}$;
    $b' \leftarrow \mathcal{B}^{\mathsf{O}_b(\cdot,\cdot)}([\boldsymbol{\Gamma}]_1)$;
    return $b = b'$

$\mathsf{O}_0([\boldsymbol{\Gamma}]_1)$
    $\boldsymbol{\alpha} \leftarrow_\$ \mathbb{Z}_p^n, [\boldsymbol{x}]_1 \leftarrow_\$ \mathcal{X}_{\mathsf{aux}} \setminus \mathcal{L}_{\mathsf{aux}}, [\boldsymbol{x}']_1 \leftarrow_\$ \mathbb{G}_1^n$;
    $R := \left(([\boldsymbol{\Gamma\alpha}]_1, [\boldsymbol{\alpha}]_1, [\boldsymbol{x}]_1, [x_i'\alpha_i]_1, [x_i'\alpha_i x_i]_1; [\boldsymbol{\alpha}]_2)\right)_{i\in[n]}$;
    return $R$

$\mathsf{O}_1([\boldsymbol{\Gamma}]_1)$
    $\boldsymbol{\alpha} \leftarrow_\$ \mathbb{Z}_p^n; [\boldsymbol{x}]_1 \leftarrow_\$ \mathcal{X}_{\mathsf{aux}} \setminus \mathcal{L}_{\mathsf{aux}}, [\boldsymbol{x}']_1 \leftarrow_\$ \mathbb{G}_1^n, \boldsymbol{u} \leftarrow_\$ \mathbb{Z}_p^n$;
    $R := \left(([\boldsymbol{\Gamma\alpha}]_1, [\boldsymbol{\alpha}]_1, [\boldsymbol{x}]_1, [u_i]_1, [x_i'\alpha_i x_i]_1; [\boldsymbol{\alpha}]_2)\right)_{i\in[n]}$;
    return $R$
```

Fig. 2. Experiment $\mathsf{Exp}^{int}_{\mathcal{B}}(\mathsf{pars})$ for the proof of smoothness in Theorem 1.

Thus, $\mathcal{A}$ is successful in breaking the soundness game iff $\mathcal{B}$ is successful in breaking the *intermediate assumption*.

We now show that the *intermediate assumption* can be reduced to the XDH problem, i.e., it is hard to distinguish the two distributions, $\{[1, a, b, ab]_1, [1]_2\}$ and $\{[1, a, u, ab]_1, [1]_2\}$ where $a, b, u \leftarrow_\$ \mathbb{Z}_p$. Let $\mathcal{D}$ be the adversary against this problem, such that given $T = \{[1, a, z, ab]_1, [1]_2\}$ it outputs 0 if $z = b$ and 1 otherwise. Given the tuple T, $\mathcal{D}$ uses $\mathcal{B}$ as a subroutine. In particular, $\mathcal{D}$ plays the role of the challenger for $\mathcal{B}$ in the experiment $\mathsf{Exp}^{int}_{\mathcal{B}}(\mathsf{pars})$ in Fig. 2 and on input $([\boldsymbol{\Gamma}]_1)$ works as follows:

1. By random self-reducibility of DDH, generate n DDH challenges $[u_i, v_i, w_i]_1$, for $i \in [1 \,..\, n]$.
2. Let $[\boldsymbol{u}]_1 = ([u_1]_1, \ldots, [u_n]_1) \in \mathbb{G}^n$, $[\boldsymbol{v}]_1 = ([v_1]_1, \ldots, [v_n]_1) \in \mathbb{G}^n$, and $[\boldsymbol{w}]_1 = ([w_1]_1, \ldots, [w_n]_1) \in \mathbb{G}^n$.
3. Choose $\boldsymbol{\alpha} \leftarrow_\$ \mathbb{Z}_p^n$ and set $R \leftarrow (([\boldsymbol{\Gamma\alpha}]_1, [\boldsymbol{\alpha}]_1, [u_i]_1, [v_i\alpha_i]_1, [\alpha_i w_i]_1; [\boldsymbol{\alpha}]_2))_{i\in[n]}$.
4. When $\mathcal{B}([\boldsymbol{\Gamma}]_1)$ calls the oracle O_b, the adversary $\mathcal{D}$ answers with R.
5. Return $\mathcal{B}$'s output.

Thus, $\mathcal{D}$ is successful in breaking the XDH problem iff $\mathcal{B}$ is successful in breaking the *intermediate assumption*. This concludes the proof. $\square$

3.1 PC-SPHF on ElGamal Ciphertexts

We design a PC-SPHF for the ElGamal language,

$$\mathcal{L}_{[m]_1} = \left\{ [\mathbf{c}]_1 = ([u]_1, [v]_1) \in \mathbb{G}_1^2 : \exists r \in \mathbb{Z}_p : ([u]_1 = r[g]_1, [v]_1 = [m]_1 + r[h]_1) \right\}.$$

The CRS $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$ contains the ElGamal encryption public key $\mathsf{pk} = [g, h]_1 \in \mathbb{G}_1^2$. With the hashing key $\mathsf{hk} = (\eta, \theta) \leftarrow_\$ \mathbb{Z}_p^2$ and the projection key $\mathsf{hp} = ([\mathsf{hp}_1]_1, [\mathsf{hp}_2]_2)$, where $[\mathsf{hp}_1]_1 = \eta[g]_1 + \theta[h]_1$, and $[\mathsf{hp}_2]_2 = [\eta, \theta]_2 \in \mathbb{G}_2^2$, and $\mathsf{aux} = [m]_1$, the hash values of the PC-SPHF are defined as follows:

$$\mathsf{H} = \mathsf{hash}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, [m]_1, [\mathbf{c}]_1) = \eta[u]_1 + \theta([v]_1 - [m]_1) \in \mathbb{G}_1$$

$$\begin{aligned}\mathsf{pH} =& \mathsf{projhash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, [m]_1, [\mathbf{c}]_1, r) = r[\mathsf{hp}_1]_1 \in \mathbb{G}_1\\ \mathsf{pcH} =& \mathsf{pchash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, [m]_1, [\mathbf{c}]_1) = [u]_1 \bullet [\mathsf{hp}_{21}]_2 + ([v]_1 - [m]_1) \bullet [\mathsf{hp}_{22}]_2 \in \mathbb{G}_T\end{aligned}$$

where we observe that $\mathsf{H} \bullet [1]_2 = \mathsf{pH} \bullet [1]_2 = \mathsf{pcH}$.

3.2 PC-SPHF on (Labeled) Cramer-Shoup Ciphertexts

We show how to extend the SPHF on (labeled) CS ciphertexts into a PC-SPHF. The CRS $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$ contains the encryption public key pk. With the hashing key $\mathsf{hk} = (\eta_1, \eta_2, \theta, \mu, \iota) \leftarrow_{\$} \mathbb{Z}_p^5$ and the projection key $\mathsf{hp} = ([\mathsf{hp}_1]_1, [\mathsf{hp}_2]_2)$, where $[\mathsf{hp}_{11}]_1 = \eta_1[g_1]_1 + \theta[g_2]_1 + \mu[h]_1 + \iota[c]_1$, and $[\mathsf{hp}_{12}]_1 = \eta_2[g_1]_1 + \iota[d]_1$, and $[\mathsf{hp}_2]_2 = [\eta_1, \eta_2, \theta, \mu, \iota]_2 \in \mathbb{G}_2^5$, and $\mathsf{aux} = [m]_1$, the hash values of the PC-SPHF are defined as follows:

$$\begin{aligned}\mathsf{H} =& \mathsf{hash}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, [m]_1, [\mathbf{c}]_1)\\ =& (\eta_1 + \xi\eta_2)[u_1]_1 + \theta[u_2]_1 + \mu([e]_1 - [m]_1) + \iota[v]_1 \in \mathbb{G}_1\\ \mathsf{pH} =& \mathsf{projhash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, [m]_1, [\mathbf{c}]_1, r) = r[\mathsf{hp}_{11}]_1 + r\xi[\mathsf{hp}_{12}]_1 \in \mathbb{G}_1\\ \mathsf{pcH} =& \mathsf{pchash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, [m]_1, [\mathbf{c}]_1)\\ =& [u_1]_1 \bullet [\mathsf{hp}_{21}]_2 + [u_1]_1 \bullet \xi[\mathsf{hp}_{22}]_2 + [u_2]_1 \bullet [\mathsf{hp}_{23}]_2 + ([e]_1 - [m]_1) \bullet [\mathsf{hp}_{24}]_2\\ &+ [v]_1 \bullet [\mathsf{hp}_{25}]_2 = [u_1]_1 \bullet [\eta_1]_2 + [u_1]_1 \bullet \xi[\eta_2]_2 + [u_2]_1 \bullet [\theta]_2\\ &+ ([e]_1 - [m]_1) \bullet [\mu]_2 + [v]_1 \bullet [\iota]_2 \in \mathbb{G}_T\end{aligned}$$

This PC-SPHF construction for labeled CS ciphertexts will be the core of constructing UC secure commitment scheme in the next section.

4 UC-Secure Commitment Scheme from PC-SPHFs

In this section, we introduce a direct application of the previous PC-SPHF on labeled CS ciphertexts to construct an efficient UC-secure commitment. Intuitively, the commit phase contains a labeled CS ciphertext. The projective hash pH will be revealed in the opening phase. The verification phase can be done by computing pcH. Finally, the simulator by having access to the trapdoor hk, computes the hash H as the simulated proof pH. Before presenting our concrete construction, we describe a generic UC-secure commitment scheme from any IND-CCA secure labeled encryption scheme with an associated PC-SPHF. Our efficient UC-secure commitment is an instantiation of this generic commitment.

4.1 Generic UC-Secure Commitment

The ideal functionality of a commitment scheme is depicted in Fig. 3. It has been proposed by Canetti and Fischlin [CF01]. Note that the functionality now takes another unique "commitment identifier" cid, which may be used if a sender

$\mathcal{F}_{\mathsf{mcom}}$, parameterized by a message space $\mathcal{M}$, interacts with adversary Sim and parties $P_1, \ldots, P_n$ as follows.

- Upon receiving $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ from P_i, where $M \in \mathcal{M}$, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, \ldots)$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously recorded, do nothing. Otherwise, record $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ and send $(\texttt{receipt}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ to P_j and Sim.
- Upon receiving $(\texttt{open}, \mathsf{sid}, \mathsf{cid})$ from P_i, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ was previously recorded then send $(\texttt{open}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ to P_j and Sim. Otherwise do nothing.
- Upon receiving $(\texttt{corrupt}, \mathsf{sid}, \mathsf{cid})$ from the adversary, send M to the adversary if there is already an entry $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$. Change the record to $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, M^*)$, if the adversary provides some M^* and $(\texttt{receipt}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ has not yet been written on P_j's output tape.

Fig. 3. Functionality $\mathcal{F}_{\mathsf{mcom}}$ for committing multiple messages

$K_{crs}(1^\lambda)$: Generate a secret and public key $(\mathsf{sk}, \mathsf{pk})$ for a labeled IND-CCA encryption scheme, set $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}} = \mathsf{pk}$. Compute $\mathsf{hk} \leftarrow \mathsf{hashkg}(\mathcal{L}_{\mathsf{aux}})$ and $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \cdot)$ and set $\mathtt{crs} := (\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{hp})$.

// Commit phase:

$\mathsf{Commit}(\mathtt{crs}, M, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: to commit to message $M \in \mathbb{G}_1$ for party P_j, upon receiving a command $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$, party P_i chooses randomness r and computes $\mathbf{c} = \mathsf{Enc}^{\mathsf{t}}_{\mathsf{pk}}(M; r)$ with $\mathsf{t} = (\mathsf{sid}, \mathsf{cid}, P_i)$ and $\mathsf{pH} \leftarrow \mathsf{projhash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, M, \mathbf{c}, r)$. P_i erases r and sends $\mathbf{c}$ to P_j and stores pH. Upon receiving $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, \mathbf{c})$ from P_i, party P_j verifies $\mathbf{c}$ is well-formed. If yes, P_j outputs $(\texttt{receipt}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$. Otherwise, P_j ignores the message.

// Opening phase:

$\mathsf{Open}(M, \mathsf{pH}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: when receiving a command $(\texttt{open}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$, party P_i reveals M and his state information pH.

// Verification phase:

$\mathsf{Ver}(\mathtt{crs}, (\texttt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c}), M, \mathsf{pH}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$: P_j computes $\mathsf{pcH} \leftarrow \mathsf{pchash}(\mathsf{hp}, \mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, M, \mathbf{c})$ and verifies pH, i.e., whether $\mathsf{pH} \bullet [1]_2 = \mathsf{pcH}$, and ignores the opening if verification fails. If verification succeeds, P_j outputs $(\texttt{open}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ iff cid has not been used with this committer previously. Otherwise, P_j also ignores the message.

Fig. 4. Generic UC-Secure Commitment from PC-SPHFs.

commits to the same receiver multiple times within a session. We assume that the combination of sid and cid is globally unique. Our generic commitment, depicted in Fig. 4, is secure in the UC framework against adaptive corruptions (assuming reliable erasure), with a common reference string for any PC-SPHF on the language of a valid ciphertext on a message M under a labeled IND-CCA-secure encryption scheme. More formally, we show the following:

Theorem 2. *The commitment scheme in Fig. 4 securely realizes $\mathcal{F}_{\mathsf{mcom}}$ in the CRS model against adaptive corruptions (assuming reliable erasure), provided that (i) $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, is an IND-CCA labeled PKE; (ii) the PC-SPHF is (t, ϵ)-sound and computationally smooth.*

For the proof we note that the simulator Sim first generates the CRS, with an encryption key pk, while knowing the decryption key sk for an IND-CCA-secure labeled encryption scheme, and the parameters for the PC-SPHF.

Proof. Intuitively, we construct an ideal-world adversary Sim that runs a black-box simulation of the real-world adversary $\mathcal{A}$ by simulating the protocol execution and relaying messages between $\mathcal{A}$ and the environment $\mathcal{Z}$. Sim proceeds as follows in experiment *IDEAL*:

- Upon the environment $\mathcal{Z}$ requires some uncorrupted party P_i to send $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ to the functionality, Sim is informed that a commitment operation took place, without knowing the committed message M. Thus, Sim selects a fake random message $M' \leftarrow_{\$} \mathbb{G}_1$, and computes an encryption $\mathbf{c}$ of $M' \in \mathbb{G}_1$. Upon P'_j outputs $(\texttt{receipt}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$, the adversary $\mathcal{A}$ is given $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$. Sim allows $\mathcal{F}_{\mathsf{mcom}}$ to proceed with the delivery of message $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ to P_j.
- If $\mathcal{Z}$ requires some uncorrupted party P_i to open a previously generated commitment $\mathbf{c}$ to some message $M \in \mathbb{G}_1$, Sim learns M from $\mathcal{F}_{\mathsf{mcom}}$ and, using the trapdoor hk of the simulated PC-SPHF hp, generates a simulated proof H that satisfies the verification algorithm Ver for the message M obtained from $\mathcal{F}_{\mathsf{mcom}}$. The internal state of P'_i is modified to be H, which is also given to $\mathcal{A}$ as the real-world opening.
- When $\mathcal{A}$ delivers $(\texttt{commit}, \mathsf{sid}', \mathsf{cid}', \mathbf{c}')$ for P'_i to P'_j, (and P'_j still has not received a commitment with cid' from P'_i), Sim proceeds as follows:
  (a) If P'_i is uncorrupted, Sim notifies $\mathcal{F}_{\mathsf{mcom}}$ that the commitment $(\mathsf{sid}', \mathsf{cid}')$ can be delivered. The $\texttt{receipt}$ message from $\mathcal{F}_{\mathsf{mcom}}$ is delivered to the dummy P_j as soon as the simulated P'_j outputs his own $\texttt{receipt}$ message.
  (b) If P_i is a corrupted party, then $\mathbf{c}'$ has to be extracted. Indeed, Sim checks if $\mathbf{c}'$ is well-formed. It uses sk corresponding to pk to decrypt $\mathbf{c}'$.
  (c) For an invalid $\mathbf{c}'$, the commitment is ignored. Otherwise, Sim receives M and sends $(\texttt{commit}, \mathsf{sid}', \mathsf{cid}', P_i, P_j, M)$ to $\mathcal{F}_{\mathsf{mcom}}$, which causes $\mathcal{F}_{\mathsf{mcom}}$ to prepare a $\texttt{receipt}$ message for P_j. The latter is delivered by Sim as soon as P'_j produces his own output.
- If $\mathcal{A}$ gets a simulated corrupted P'_i to correctly open a commitment $(\texttt{commit}, \mathsf{sid}', \mathsf{cid}', \mathbf{c}')$ to M', Sim compares M' to M that was previously extracted from $\mathbf{c}'$ and aborts if $M \neq M'$. Otherwise, Sim sends $(\texttt{open}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ on behalf of P_i to $\mathcal{F}_{\mathsf{mcom}}$. If $\mathcal{A}$ provides an incorrect opening, Sim ignores this opening.
- If $\mathcal{A}$ decides to corrupt some party P'_i, Sim corrupts the corresponding party P_i in the ideal world and obtains all his internal information. In order to match the received opening information of P_i, Sim modifies all opening information about the unopened commitments generated by P'_i. This modified internal information is given to $\mathcal{A}$. For each commitment intended for P_j but

for which P_j did not receive $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$, the newly corrupted P_i' is allowed to decide what the committed message will be. A new message M is thus provided by $\mathcal{A}$ and Sim informs $\mathcal{F}_{\mathsf{mcom}}$ that M supersedes the message chosen by P_i before his corruption.

We consider a sequence of *hybrid* games between the *real* and *ideal* worlds and show that the commitment scheme emulates the ideal functionality against adaptive corruptions with erasures. This is a general approach which one can follow to prove the security of a commitment scheme in the UC model. The games starts from the real game, adversary $\mathcal{A}$ interacts with real parties, and ends up with the ideal game. In the ideal game, we build Sim that interfaces between adversary $\mathcal{A}$ and ideal functionality $\mathcal{F}_{\mathsf{mcom}}$.

Game_0*:* This is called *real game* ($\textsc{Hybrid}^{\mathcal{F}_{\mathsf{mcom}}}$), which corresponds to the real world in the CRS model. In this game, the real protocol is executed between committer P_i and receiver P_j. Environment $\mathcal{Z}$ adaptively chooses the input for honest committer P_i and receives output of the honest parties. Naturally, there is an adversary $\mathcal{A}$ that attacks the real protocol in the real world, i.e., it can corrupt some parties and see all flows from parties. In the case of corruption, $\mathcal{A}$ can read the current inner state of the corrupted party and also can fully control it. In this game, environment $\mathcal{Z}$ can control adversary $\mathcal{A}$ and see exchanged messages among all honest parties, and all of $\mathcal{A}$'s interactions with other parties.

Game_1*:* In the setup phase of this game, simulator Sim chooses $\mathsf{hk} \leftarrow_{\$} \mathbb{Z}_p^n$, and generates $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathtt{crs})$, $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$ and sets $\mathtt{crs} = (\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{hp})$. In the commit phase, upon receiving a query $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ from $\mathcal{Z}$, corrupted party P_i' computes $\mathbf{c}$ using the labeled IND-CCA encryption scheme and sends $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$ to receiver P_j. In the commit phase, after receiving $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, \mathbf{c})$, Sim decrypts and stores $M' = \mathsf{Dec}_{\mathsf{sk}}^{\mathsf{t}}(\mathbf{c})$, where $\mathsf{t} \leftarrow (\mathsf{sid}, \mathsf{cid}, P_i)$. In the opening phase, when P_i' successfully opens to message M, simulator Sim outputs $(\texttt{reveal}, \mathsf{t}, M')$ to environment $\mathcal{Z}$.

Lemma 1. *If $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, the labeled PKE is IND-CCA secure, and PC-SPHF is computationally smooth, the output of $\mathcal{Z}$ in Game_0 and Game_1 is computationally indistinguishable.*

Proof. In Game_1, we observed that after P_i' opened commitment to message M, Sim reveals decrypted message M'. Suppose that bad defines the case that sender P_i' successfully opened message M but $M \neq M'$. Now, the claim states that if bad happens, that is the smoothness of PC-SPHF is broken; which it happens with a negligible probability. More precisely, let $M' \neq M$. Then, in the opening phase it uses $\pi = \mathsf{pH}$ and successfully opens the commitment to M'. The case impels party P_i' comes up with a valid proof for PC-SPHF where it turns that smoothness of PC-SPHF is broken.

Hence, the case bad happens only with a negligible probability and two games Game_0 and Game_1 are computationally indistinguishable in a view of $\mathcal{Z}$. $\square$

Game_2*:* This game is the same with Game_1, the only difference is that simulator Sim modifies the simulation of an honest sender P_i. In the commit phase, after receiving a query $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ from $\mathcal{Z}$, the simulator Sim computes $[\mathbf{c}]_1$ and sends $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$ to P_j (note that we assume that Sim knows M). In the opening phase, upon getting an Open query, Sim uses simulates a proof π by computing the hash value H. Finally, simulator Sim outputs $(\mathtt{reveal}, \pi, M)$ to the environment $\mathcal{Z}$.

Lemma 2. *The output of $\mathcal{Z}$ in* Game_1 *and* Game_2 *is computationally indistinguishable.*

Proof. The proof of this lemma is straightforward and lies in the soundness property of the PC-SPHF. Following the mentioned property, $\mathcal{Z}$'s views are statistically close in both games. □

Game_3*:* In this game, similar to last one, Sim again modifies the simulation of an honest sender P_i. But, in the commitment phase, Sim commits to message $M = [0]_1 = 1$. Accurately, upon receiving a query $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ from $\mathcal{Z}$, the simulator Sim computes $\mathbf{c}$ with $M = 1$ and sends $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$ to P_j. In the opening phase, similar to Game_2, upon getting Open query, Sim simulates a π by computing the hash value H using the trapdoor keys (hk, τ). Finally, simulator Sim outputs $(\mathtt{reveal}, \pi, M)$ to the environment $\mathcal{Z}$.

Lemma 3. *Let $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$, a labeled PKE, be IND-CCA. Then, the $\mathcal{Z}$'s view in* Game_2 *is computationally indistinguishable from* Game_3*.*

Proof. From the description of Game_3, we observe that the only difference with Game_2 is that, in this game the simulator Sim computes $\mathbf{c}$ with 1 instead of M. Actually, by having the trapdoor keys of the PC-SPHF, Sim can commit 1 in the commitment phase, and opens to M in the opening phase. Also by considering the IND-CCA property, we can say that Game_2 and Game_3 are statically close. As analyzed in Game_1, $\Pr[\mathsf{bad}_0] = \Pr[\mathsf{bad}_1] = \mathsf{negl}(\lambda)$. Similarly, in analysis of Game_2, we observed that Game_1 and Game_2 are statistically indistinguishable, so $\Pr[\mathsf{bad}_2] = \Pr[\mathsf{bad}_1] = \mathsf{negl}(\lambda)$. As already Game_2 and Game_3 are statically close, we conclude that the $\Pr[\mathsf{bad}_2] \approx \Pr[\mathsf{bad}_3] = \mathsf{negl}(\lambda)$. □

Game_4*:* This game corresponds to the *ideal world* in the CRS model. In the ideal world, there exists an ideal functionality $\mathcal{F}_{\mathsf{mcom}}$ and the task of the honest parties in the ideal world simply convey inputs from environment $\mathcal{Z}$ to the ideal functionalities and vice versa. In ideal word, the ideal honest parties interact only with the environment $\mathcal{Z}$ and the ideal functionalities. In this game, the ideal-world adversary Sim proceeds as follows:

- *Initialization step:* Sim chooses $\mathsf{hk} \leftarrow_{\$} \mathbb{Z}_p^n$, and generates $\mathsf{hp} \leftarrow \mathsf{projkg}(\mathsf{hk}, \mathtt{crs})$, $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$ and sets $\mathtt{crs} = (\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{hp})$. In addition, it chooses a collision-resistant hash function H.
- *Simulating the communication with $\mathcal{Z}$:* Every input value that Sim receives from $\mathcal{Z}$ is written on $\mathcal{A}$'s input tape (as if coming from $\mathcal{Z}$) and vice versa.

- *Simulating the commit phase when committer P_i is honest:* Upon receiving the receipt message $(\texttt{receipt}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ from $\mathcal{F}_{\mathsf{mcom}}$, Sim computes $\mathbf{c}$ with $M = [0]_1$ and sends $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$ to P_j.
- *Simulating the opening phase when P_i is honest:* Upon receiving input $(\texttt{open}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ from $\mathcal{F}_{\mathsf{mcom}}$, simulator computes proof $\pi = \mathsf{pH}$, and sends $(\mathsf{sid}, \mathsf{cid}, P_i, M, \pi)$ to P_j.
- *Simulating adaptive corruption of P_i after the commit phase but before the opening phase:* When P_i is corrupted, Sim can immediately read ideal P_i's inner state and obtain M. Then, Sim produces $\mathbf{c}$ as in the case of the opening phase when P_i is honest and outputs $(\texttt{reveal}, \mathsf{sid}, \mathsf{cid}, \mathbf{c}, \pi, M)$ to the P_j.
- *Simulating the commit phase when committer $\hat{P}_i$ is corrupted and the receiver P_j is honest:* After receiving $(\texttt{commit}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$ from $\hat{P}_i$ controlled by $\mathcal{A}$ in the commit phase, Sim decrypts $M' = \mathsf{Dec}^{\mathsf{t}}_{\mathsf{sk}}(\mathbf{c})$, where $\mathsf{t} \leftarrow (\mathsf{sid}, \mathsf{cid}, P_i)$ and sends $(\texttt{com}, \mathsf{t}, P_j, M')$ to $\mathcal{F}_{\mathsf{mcom}}$.
- *Simulating the opening phase when committer $\hat{P}_i$ is corrupted and receiver P_j is honest:* Upon receiving, $(\texttt{open}, \mathsf{sid}, \mathsf{cid}, M, \pi)$ from corrupted committer $\hat{P}_i$ controlled by $\mathcal{A}$, as it expects to send to P_j, Sim sends $(\texttt{open}, \mathsf{sid}, \mathsf{cid})$ to $\mathcal{F}_{\mathsf{mcom}}$. ($\mathcal{F}_{\mathsf{mcom}}$ follows its codes: If a tuple $(\mathsf{sid}, \mathsf{cid}, P'_i, P_j, M')$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously stored by $\mathcal{F}_{\mathsf{mcom}}$, $\mathcal{F}_{\mathsf{mcom}}$ sends $(\texttt{reveal}, \mathsf{sid}, \mathsf{cid}, P'_i, M')$ to ideal receiver P_j and Sim. Then, ideal receiver P_j convey it to $\mathcal{Z}$).
- *Simulating adaptive corruption of P_j after the commit phase but before the opening phase:* When P_j is corrupted, Sim simply outputs $(\texttt{reveal}, \mathsf{sid}, \mathsf{cid}, \mathbf{c})$.

By construction, Game_4 is identical to the Game_3. □

4.2 Efficient Instantiation

Let us now instantiate this generic commitment with the labeled CS encryption scheme and our PC-SPHF on labeled CS ciphertexts. The resulting scheme is depicted in Fig 5. The commitment consists of 4 elements in $\mathbb{G}_1$ and the opening of one element in $\mathbb{G}_1$, which, to the best of our knowledge, makes it the most efficient non-interactive UC-secure commitment scheme.

For concrete figures, we compare our instantiation to the most efficient instantiation under the plain DDH assumption in [ABP17]. For the comparison let us assume a type 3 bilinear group with a desired security level of 128 bit. A popular choice are Baretto-Naehrig (BN) or Barreto-Lynn-Scott (BLS) curves. A conservative estimate for this security level yields elements in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ of size $2 \cdot 384$, $4 \cdot 384$ and $12 \cdot 384$ bits for BN and BLS12 and $2 \cdot 320$, $4 \cdot 320$ and $24 \cdot 320$ for BLS24 (without point compression) respectively [MSS16]. Assuming that we use elliptic curves over prime fields to instantiate the plain DDH setting, elements of $\mathbb{G}$ will have at least $2 \cdot 256$ bits (without point compression) when targeting 128 bit security. Consequently, assuming point compression is used in both schemes, the commitment and opening size of the UC-commitment in [ABP17] is 1799 and 512 bits respectively. Our UC-secure commitment has a commitment and opening size of 1284 and 321 bit respectively, improving [ABP17] by about 30%. Furthermore, compared to the most efficient construction in bilinear groups (i.e., [ABB+13]), we obtain an improvement in the opening size of a factor 4.

$K_{crs}(1^\lambda)$: Compute and return hp and the CRS $\mathtt{crs} = (\mathsf{pk}, \mathsf{hp})$. // similar to Fig 4

// Commit phase:

$\mathsf{Commit}(\mathtt{crs}, M, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$: to commit to message $M = [m]_1 \in \mathbb{G}_1$ for R, upon receiving a command $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, M)$, C does the following,

(a) Choose $r \leftarrow_\$ \mathbb{Z}_p$ and compute $[\mathbf{c}]_1 = [u_1, u_2, e, v]_1 \in \mathbb{G}_1^4$ where $[u_1]_1 = r[g_1]_1$, $[u_2]_1 = r[g_2]_1$, $[e]_1 = [m]_1 + r[h]_1$, $[v]_1 = r([c]_1 + \xi[d]_1)$, where $\xi = H(\mathsf{t}, [u_1]_1, [u_2]_1, [e]_1)$ and $\mathsf{t} = (\mathsf{sid}, \mathsf{cid}, P_i)$.;

(b) Compute $\mathsf{pH} = r[\mathsf{hp}_{11}]_1 + r\xi[\mathsf{hp}_{12}]_1 \in \mathbb{G}_1$;

(c) Send $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, [\mathbf{c}]_1)$ to R. Securely erase randomness r and store $\pi = \mathsf{pH}$.

(d) Upon receiving $(\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, [\mathbf{c}]_1)$, R checks that $[\mathbf{c}]_1 \in \mathbb{G}_1^4$. If yes, outputs $(\mathtt{receipt}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$ and stores $[\mathbf{c}]_1$. Otherwise, R ignores it.

// Opening phase:

$\mathsf{Open}([m]_1, \mathsf{pH}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$: when receiving a command $(\mathtt{open}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, [m]_1)$, party C reveals $[m]_1$ and his state information $\pi = \mathsf{pH}$.

// Verification phase:

$\mathsf{Ver}(\mathtt{crs}, (\mathtt{commit}, \mathsf{sid}, \mathsf{cid}, [\mathbf{c}]_1), [m]_1, \mathsf{pH}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R})$: computes $\mathsf{pcH} = [u_1]_1 \bullet [\eta_1]_2 + [u_1]_1 \bullet \xi[\eta_2]_2 + [u_2]_1 \bullet [\theta]_2 + ([e]_1 - [m]_1) \bullet [\mu]_2 + [v]_1 \bullet [\iota]_2 \in \mathbb{G}_T$ and verifies $\pi \bullet [1]_2 = \mathsf{pcH}$. R ignores the opening if the verification fails. If verification succeeds, R outputs $(\mathtt{open}, \mathsf{sid}, \mathsf{cid}, \mathsf{C}, \mathsf{R}, [m]_1)$ iff cid has not been used with this committer previously. Otherwise, R also ignores the message.

Fig. 5. UC-Secure commitment from PC-SPHF for the labeled CS encryption scheme.

5 Anonymous Credential System-Based Message Transmission

Anonymous credentials (ACs) were introduced in the seminal work of Chaum [Cha86], and allow users to anonymously authenticate to a variety of services. Typical use-cases of ACs involve three main parties, users, authorities (organizations), and verifiers (servers). Each user can receive credentials (which can be a set of attributes) from authorities, and register pseudonyms with authorities and verifiers. Then users can prove to verifiers that a subset of their attributes verifies some policy P. The pseudonyms associated to the identity of the user should be unlinkable to its exact identity, i.e., another entity should not be able to check whether two pseudonyms are associated with the same identity. Due to their wide range of real-world applications, anonymous credentials have received a lot of attention from the cryptographic community, e.g., [Cha86,CL01,CL03,CL04,BCC+09,CKL+16,FHS19].

In this section, we revisit the use of anonymous credentials for message recovery proposed in [BC16]. Similar to [BC16], we present a constant-size, round-optimal protocol that allows to use an anonymous credential to retrieve a message without revealing the identity of the receiver in a UC secure way, but more efficient than the one proposed in [BC16]. We follow the scenario of [BC16], and assume that different organization issue credentials to users. The full construction is shown in Fig. 6.

CRS generation. Run $K_{crs}(1^\lambda)$ and generate hp and the CRS $\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}$ and set $\mathtt{crs} = (\mathtt{crs}_{\mathcal{L}_{\mathsf{aux}}}, \mathsf{hp})$. // similar to Section 3.2

Pre-flow. The server generates a key pair $(\mathsf{pk}, \mathsf{sk})$ of a CPA secure encryption scheme. It stores sk and sends pk to user U.

Credential use by user U_i. The user U_i does the following:

(a) Choose a random value J. Compute $J_F \leftarrow F(J)$ and $\boldsymbol{c}_J = \mathsf{Enc}^{cpa}_{\mathsf{pk}}(J)$, as an encryption of J under pk.

(b) Choose $r \leftarrow_\$ \mathbb{Z}_p$ and compute the CS ciphertext $[\mathbf{c}]_1 = [u_1, u_2, e, v]_1 \in \mathbb{G}_1^4$. Compute $\mathsf{pH} = (r[\mathsf{hp}_{11}]_1 + r\xi[\mathsf{hp}_{12}]_1) \in \mathbb{G}_1$ and $\boldsymbol{c}_{\mathsf{Cred}} = \mathsf{Enc}^{cpa}_{\mathsf{pk}}([\mathsf{Cred}_i]_1)$.

(c) Send $(\boldsymbol{c}_J, [\mathbf{c}]_1, \boldsymbol{c}_m)$ to the server. Store pH (as the new witness) and J_F, and securely erase J and the randomnesses used in the encrypting processes.

Database input Doc with policy P. When receiving the messages $(\boldsymbol{c}_J, [\mathbf{c}]_1, \boldsymbol{c}_m)$ from U_i , the server S decrypts $J \leftarrow \mathsf{Dec}^{cpa}_{\mathsf{sk}}(\boldsymbol{c}_J)$, $[m]_1 \leftarrow \mathsf{Dec}^{cpa}_{\mathsf{sk}}(\boldsymbol{c}_m)$, and computes $F(J)$ and does the following:

(a) Compute $\mathsf{pcH} \in \mathbb{G}_T$. Choose $s \leftarrow_\$ \mathbb{Z}_p$ and compute $\mathsf{H}_S = s \cdot \mathsf{pcH} \in \mathbb{G}_T$ and $\mathsf{hp}_S = s[1]_2$. Compute $\boldsymbol{c}_S = F(J) \oplus \mathsf{H}_S \oplus \mathsf{Doc}$.

(b) Erase everything except $(\boldsymbol{c}_S, \mathsf{hp}_S)$ and send them over a secure channel.

Data recovery. When receiving the tuple $(\boldsymbol{c}_S, \mathsf{hp}_S)$, the user U_i, first computes $\mathsf{pH}_D = \mathsf{pH} \bullet \mathsf{hp}_S \in \mathbb{G}_T$, and then retrieves $\mathsf{Doc} = \boldsymbol{c}_S \oplus J_F \oplus \mathsf{pH}_D$.

Fig. 6. UC-Secure Anonymous Credential System-Based Message Transmission from an PC-SPHF.

For the security analyzing, we first describe the ideal functionality $\mathcal{F}_{\mathsf{ac}}$ for Anonymous Credential-Based Message Transmission proposed by [BC16]. It is depicted in Fig. 7. The user U_i received Doc form the server S when her credentials Cred comply with the policy P.

Theorem 3. *The Anonymous Credential System-Based Message Transmission protocol described in Fig. 6 is UC secure in the presence of adaptive adversaries, assuming reliable erasures and authenticated channels.*

Proof. From a high level point of view, in the case of adaptive corruptions and for the simulating procedure we use an extractable and equivocable commitment, which allows the simulator Sim to simply open the commitment to any message (credential). Intuitively, the equivocability property of the commitment enables

$\mathcal{F}_{\mathsf{ac}}$, parameterized by a message space $\mathcal{D} \in \{0,1\}^{\lambda}$, interacts with adversary Sim and parties $P_1, \ldots, P_n$ as follows.

- Upon receiving $(\texttt{send}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, \mathsf{Doc})$ from P_i, where $\mathsf{Doc} \in \mathcal{D}$, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, \mathsf{Doc})$ with the same $(\mathsf{sid}, \mathsf{cid})$ was previously recorded, do nothing. Otherwise, record $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, M)$ and reveal $(\texttt{send}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ to Sim.
- Upon receiving $(\texttt{receive}, \mathsf{sid}, \mathsf{cid}, \mathsf{Cred})$ from P_j, proceed as follows: if a tuple $(\mathsf{sid}, \mathsf{cid}, P_i, P_j, \mathsf{Doc})$ was previously recorded then send $(\texttt{receive}, \mathsf{sid}, \mathsf{cid}, P_i, P_j)$ to Sim, and send $(\texttt{receive}, \mathsf{sid}, \mathsf{cid}, P_i, P_j, \mathsf{Doc}')$ to P_J, where $\mathsf{Doc}' = \mathsf{Doc}$ if the credentials comply with the policy P, and $\mathsf{Doc}' = \perp$ otherwise. Ignore further $\texttt{receive}$ messages with the same cid from P_j.

Fig. 7. Functionality $\mathcal{F}_{\mathsf{ac}}$ for Ideal Functionality for Anonymous Credential-Based Message Transmission

Sim to adapt the incorrect credential and the used randomness such that they seem to be in the language. By extractability property, when simulating the sever, Sim knows whether it has to send the correct message. Recall that by adaptive corruption we mean the adversary $\mathcal{A}$ is able to corrupt any player at any time during the execution of the protocol. Notice that the simulator Sim first generates the CRS $\mathtt{crs}$ and the PC-SPHF parameters. As usual, we can construct an ideal-world adversary Sim that runs a black-box simulation of the real-world adversary $\mathcal{A}$ by simulating the protocol execution and relaying messages between $\mathcal{A}$ and the environment $\mathcal{Z}$. Sim proceeds as follows in experiment *IDEAL*. The sketch of the proof is as follows,

- In pre-flow phase, upon receiving the $\texttt{send}$ query from $\mathcal{F}_{\mathsf{ac}}$, Sim generates a key pair $(\mathsf{pk}, \mathsf{sk})$ (he knows it from an honest sender has sent a pre-flow).
- Upon receiving $\texttt{receive}$query from $\mathcal{F}_{\mathsf{ac}}$, by using an equivocable commitment, Sim computes the tuple $(\mathsf{pH}, [\mathsf{c}]_1, \boldsymbol{c}_m)$ with label $(\mathsf{sid}, \mathsf{cid}, P_i, P_j)$ and a ciphertext $\boldsymbol{c}_J \leftarrow \mathsf{Enc}^{cpa}_{\mathsf{pk}}(J)$ where J_F is a random value. Note that Sim already has received a pre-flow pk from an honest or a corrupted sender.
- If $\mathcal{Z}$ requires uncorrupted server S who received the tuple $(\boldsymbol{c}_J, [\mathsf{c}]_1, \boldsymbol{c}_{\mathsf{Cred}})$ from a corrupted U_i, Sim decrypts the ciphertexts $\boldsymbol{c}_J$, and $\boldsymbol{c}_{\mathsf{Cred}}$, and obtains J and $[\mathsf{Cred_i}]_1$. Sim, and then computes $F(J)$. It extracts the committed values (Cred) and check if it is correct and sends $\texttt{receive}$ to $\mathcal{F}_{\mathsf{ac}}$.
- When $\mathcal{Z}$ requires uncorrupted user U_i who received the tuple $(\boldsymbol{c}_S, \mathsf{hp}_S)$ from a corrupted S, Sim computes pH_D and obtains Doc and uses this value in a $\texttt{send}$ query to $\mathcal{F}_{\mathsf{ac}}$.
- In the case of an honest server, when Sim receives a $\texttt{receive}$ query and Doc from $\mathcal{F}_{\mathsf{ac}}$, it sends Doc to the corrupted user.
- When $\mathcal{Z}$ requires uncorrupted server S who is interacting with an uncorrupted user U_i, Sim sets $\mathsf{Doc} = 0$ and choose J_F randomly instead of computing it correctly by $F(J)$. It computes the commitment $(\boldsymbol{c}_J, [\mathsf{c}]_1, \boldsymbol{c}_{\mathsf{Cred}})$.

In case of corruption afterwards, due to equivocability property of the commitment the value J_F can be be adapted during the simulation such that it gives the message Doc received by the user in case his credentials comply with the policy. $\square$

6 Open Problem

In [KV09], Katz and Vaikuntanathan constructed the first (approximate) SPHF for a lattice-based language: the language of ciphertexts of some given plaintext for an LWE-based IND-CCA encryption scheme. Later, by using harmonic analysis, Benhamouda et. al [BBDQ18] improved the Katz-Vaikuntanathan construction, where the construction is over a tag-based IND-CCA encryption scheme a la Micciancio-Peikert [MP12]. An interesting open question is the construction of PC-SPHFs for the class of lattice-based languages.

Acknowledgments. This work was supported by the EU's Horizon 2020 ECSEL Joint Undertaking project SECREDAS under grant agreement n°783119, the Austrian Science Fund (FWF) and netidee SCIENCE project PROFET (grant agreement P31621-N38) and the Estonian Research Council grant PRG49.

References

ABB$^+$13. Michel Abdalla, Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, and David Pointcheval. SPHF-friendly non-interactive commitments. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 214–234. Springer, Heidelberg, December 2013.

ABL$^+$19. Behzad Abdolmaleki, Karim Baghery, Helger Lipmaa, Janno Siim, and Michal Zajac. DL-extractable UC-commitment schemes. In Robert H. Deng, Valérie Gauthier-Umaña, MartÃŋn Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 385–405. Springer, Heidelberg, June 2019.

ABP17. Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Removing erasures with explainable hash proof systems. In Serge Fehr, editor, *PKC 2017, Part I*, volume 10174 of *LNCS*, pages 151–174. Springer, Heidelberg, March 2017.

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth projective hashing for conditionally extractable commitments. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 671–689. Springer, Heidelberg, August 2009.

BBC$^+$13. Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 449–475. Springer, Heidelberg, August 2013.

BBC$^+$15. Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. Cryptology ePrint Archive, Report 2015/188, 2015. `http://eprint.iacr.org/2015/188`.

BBDQ18. Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 644–674. Springer, Heidelberg, March 2018.

BC16. Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *LNCS*, pages 339–369. Springer, Heidelberg, December 2016.

BCC$^+$09. Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, Heidelberg, August 2009.

BPV12. Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 94–111. Springer, Heidelberg, March 2012.

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

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

Cha86. David Chaum. Showing credentials without identification: Signatures transferred between unconditionally unlinkable pseudonyms. In Franz Pichler, editor, *EUROCRYPT'85*, volume 219 of *LNCS*, pages 241–244. Springer, Heidelberg, April 1986.

CKL$^+$16. Jan Camenisch, Stephan Krenn, Anja Lehmann, Gert Læssøe Mikkelsen, Gregory Neven, and Michael Østergaard Pedersen. Formal treatment of privacy-enhancing credential systems. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 3–24. Springer, Heidelberg, August 2016.

CL01. Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, Heidelberg, May 2001.

CL03. Jan Camenisch and Anna Lysyanskaya. A signature scheme with efficient protocols. In Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, editors, *SCN 02*, volume 2576 of *LNCS*, pages 268–289. Springer, Heidelberg, September 2003.

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

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, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 45–64. Springer, Heidelberg, April / May 2002.

EHK$^+$13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

ElG84. Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 10–18. Springer, Heidelberg, August 1984.

FHS19. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *J. Cryptology*, 32(2):498–546, 2019.

FLM11. Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 468–485. Springer, Heidelberg, December 2011.

JR13. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.

KV09. Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 636–652. Springer, Heidelberg, December 2009.

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 293–310. Springer, Heidelberg, March 2011.

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Heidelberg, April 2012.

MSS16. Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with assessing the impact of NFS advances on the security of pairing-based cryptography. In *Paradigms in Cryptology - Mycrypt 2016*, pages 83–108, 2016.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.