Practical Witness Encryption for Algebraic Languages Or How to Encrypt Under Groth-Sahai Proofs

David Derler and Daniel Slamanig

IAIK, Graz University of Technology, Austria
{david.derler|daniel.slamanig}@tugraz.at

Abstract. Witness encryption (WE) is a recent powerful encryption paradigm, which allows to encrypt a message using the description of a hard problem (a word in an **NP**-language) and someone who knows a solution to this problem (a witness) is able to efficiently decrypt the ciphertext. Recent work thereby focuses on constructing WE for **NP** complete languages (and thus **NP**). While this rich expressiveness allows flexibility w.r.t. applications, it makes existing instantiations impractical. Thus, it is interesting to study practical variants of WE schemes for subsets of **NP** that are still expressive enough for many cryptographic applications.

We show that such WE schemes can be generically constructed from smooth projective hash functions (SPHFs). In terms of concrete instantiations of SPHFs (and thus WE), we target languages of statements proven in the popular Groth-Sahai (GS) proof framework. This allows us to provide a novel way to encrypt. In particular, encryption is with respect to a GS proof and efficient decryption can only be done by the respective prover. The so obtained constructions are entirely practical and only require standard assumptions such as DLIN or DDH.

To illustrate our techniques, we propose an elegant fully ad-hoc solution to the following scenario. Assume a whistleblower, say Edwarda, wants to leak authoritative secrets while staying anonymous. Therefore, she signs the leaked documents using a ring signature, which hides her identity unconditionally among other carefully selected people in a potentially huge ad-hoc group. We demonstrate how to encrypt a message using *only* a ring signature such that solely the anonymous signer can decrypt and read the reply.

Keywords: Witness encryption, smooth projective hash functions, Groth-Sahai proofs, ring signatures, ring encryption, leaking secrets.

1 Introduction

Witness encryption (WE) is a recent powerful encryption paradigm introduced by Garg et al. [GGSW13]. In WE, an encryption scheme is defined for some **NP**-language L with witness relation R so that $L = \{x \mid \exists\ w : R(x, w) = 1\}$. The

The authors have been supported by EU H2020 project PRISMACLOUD, grant agreement n°644962.

encryption algorithm takes an alleged word x from L (instead of an encryption key) and a message m and produces a ciphertext c. Using a witness w such that $R(x, w) = 1$, anyone can decrypt c to obtain m. Decryption only works if $x \in L$ and a ciphertext c hides m if c has been computed with respect to some $x \notin L$.

Constructions of WE. The first construction of WE for any language in **NP** in [GGSW13] has been for the **NP**-complete problem *exact cover* and uses approximate multilinear maps (MLMs). Later, Gentry et al. [GLW14] introduced the concept of positional WE, which allows to prove the aforementioned construction secure. In [GGH+13], Garg et al. showed that indistinguishability obfuscation implies WE. Goldwasser et al. proposed the stronger notion of *extractable* WE in [GKP+13]. While the security for WE is only with respect to $x \notin L$, extractable WE requires that any successful adversary against semantic security of the WE, given an encryption with respect to x, implies the existence of an extractor that extracts a witness w to $x \in L$. Thereby, the adversary and the extractor additionally get an auxiliary input. Garg et al. [GGHW14] have shown that under the assumption that special-purpose obfuscation exists, extractable WE for all languages in **NP** cannot exist.[1] Zhandry [Zha16] introduced the concept of witness PRFs, which essentially generalizes WE. Zhandry also proposes (CCA secure) *reusable* WE, which introduces an additional global setup and thus allows to reuse certain parameters. This drastically reduces the size of ciphertexts in WE schemes. We observe that our generic constructions of WE bear similarities to how WE is constructed from witness PRFs. Yet, Zhandry aims at building witness PRFs for any **NP**-language, where we aim at practical instantiations. All these constructions build upon MLMs and/or obfuscation and are thus far from being practical. To this end, Abusalah et al. [AFP16] very recently introduced the notion of *offline* WE as a step towards more practical WE. They split encryption into an expensive offline phase and a much more efficient online phase, which allows them to achieve practical efficiency for the online part. Nevertheless, the offline part and the decryption still requires obfuscation and thus cannot be considered to be practical. Besides imposing a huge computational overhead, MLM and obfuscation are still in a "break-repair" state and it is currently unknown if one can come up with candidate constructions being secure under well established assumptions.

Restricting Languages. In concurrent and independent work, Faonio et al. [FNV15] introduced the concept of predictable arguments of knowledge (PAoK). They are one-round interactive protocols in which the verifier generates a challenge and can at the same time predict the prover's answer to that challenge. Faonio et al. show that PAoKs are equivalent to extractable WE [GKP+13]. Regarding concrete instantiations of PAoKs (and thus extractable WE), they show how to construct PAoKs from extractable hash proof systems (Ext-HPS) as defined by Wee in [Wee10]. Although their approach to constructing WE can thus be seen as related to our approach, firstly ours is conceptually simper and secondly the languages covered by Ext-HPSs are very basic and very restricted, i.e., [Wee10] presents two instantiations; one for the iterated squaring relation and one for

[1] Even if such special-purpose obfuscation exists, this does not rule out that extractable WE for a sufficiently large interesting subset of **NP** exists.

the Diffie Hellman relation. It is also not clear if efficient instantiations for more expressive languages can be found. We also note that due to the lack in expressiveness of Ext-HPS as used in [FNV15], their constructions are not suitable for what we are targeting at. Earlier work on (private) conditional oblivious transfer [COR99, JL09] can be viewed as as an interactive version of (extractable) WE for very specific and restricted languages not suitable for achieving our goals. Finally, [GGSW13] mentioned along the lines that earlier work on SPHFs can be interpreted as establishing the existence of WE for certain restricted languages and an informal sketch of a construction of WE from SPHFs was recently given in [ABP15].

Applications of WE. WE in general extends the scope of encryption as it allows to encrypt a message using the description of a hard problem and only someone who knows a solution to this problem is able to decrypt. WE is thus intuitively related to time-lock puzzles [RSW96] and WE indeed has been used to realize a related concept denoted as time-lock encryption, i.e., a method to encrypt a message such that it can only be decrypted after a certain deadline has passed, but then very efficiently and by everyone. An approach to realize such schemes from WE and so called computational reference clocks has been proposed by Jager in [Jag15]. Liu et al. [LKW15] also propose to use their WE construction for time-lock encryption based on the Bitcoin protocol. Bellare and Hoang [BH15] proposed to use WE to realize asymmetric password-based encryption, where the hash of a password can be used to encrypt a message (acting as a public key) and only the knowledge of the respective password allows decryption. Moreover, it has already been shown in the seminal work [GGSW13] that WE can be used to construct identity-based encryption (IBE) [BF01] as well as attribute-based encryption (ABE) [SW05] for circuits.

Motivation. While having WE schemes that support *all languages* in **NP** is appealing, it is the main source of inefficiency. We aim to make WE practical, but in contrast to offline WE as introduced in [AFP16] we focus on all aspects, i.e., encryption and decryption, to be efficient. Our approach to improving the efficiency is by restricting the class of supported languages from any **NP**-language to languages that are expressive enough to cover many problems encountered in cryptographic protocol design. In particular, we aim at *algebraic languages defined over bilinear groups*. Such languages are very relevant for the design of cryptographic protocols as statements in these languages cover statements that can be proven in a zero-knowledge (or witness indistinguishable) fashion using the Groth-Sahai (GS) non-interactive proof framework [GS08]. As we will see soon, our techniques yield a novel way of encryption, where one can encrypt messages with respect to a GS proof so that only the prover, i.e., the party that computed the respective proof, can decrypt. We assume that there are many interesting applications that could benefit from our technique. As an example, we apply our techniques to elegantly solve the problem discussed below.

An Application. Let us assume that a whistleblower, say Edwarda, wants to leak a secret to some journalist in a way that the journalist will have a high level of confidence that the information indeed comes from an insider. Thereby, the whistleblower has a strong interest in staying anonymous, i.e., identifying her

could lead to severe consequences. To solve this dilemma, Rivest et al. [RST01] came up with an elegant cryptographic primitive called ring signature. In such a signature scheme, a signature hides the identity of the signer unconditionally among other people in an ad-hoc group (the so called ring) selected by the signer without requiring their approval or assistance. Consequently, a whistleblower can choose a set of potential signers, e.g., other insiders, that do not even need to be aware of the fact that they are included into the ring. Thus, the whistleblower can convince the journalist that there is strong evidence that the exposed information is indeed authentic.

Given such a ring signature, an immediate question is how to privately reply to the unknown sender. The arguably most elegant solution would be a fully ad-hoc solution, i.e., one that does not require to modify the signing algorithm. In particular, we are after a solution that directly uses *only* a given ring signature to encrypt the reply to the whistleblower, while guaranteeing that only the whistleblower who has produced the ring signature can decrypt the reply.

Our Contribution. The contributions in this paper are as follows.

- We provide a generic construction of WE from SPHFs and prove that if there exists an SPHF for a language L, then there exists an adaptively sound WE scheme for language L. Thereby, we define WE to provide an additional setup algorithm as also done in [AFP16, Zha16], since this notion makes the schemes more efficient and more convenient to use in protocol design.
- Using well known techniques such as universal hashing and secure symmetric encryption schemes, we obtain a WE scheme for messages of arbitrary length.
- We present practical instantiations of our generic approach to WE for algebraic languages in the bilinear group setting. We, thereby, achieve compatibility with statements from the GS proof system. Besides being practically efficient, our constructions only require standard assumptions (i.e., DLIN).[2]
- We present an approach to use our WE construction for GS statements to elegantly encrypt messages with respect to NIZK/NIWI proofs for statements in the frequently used GS proof system so that only the one who computed the proof can decrypt. This yields a novel way of encryption.
- To illustrate the aforementioned concept, we demonstrate how to use our techniques in the context of whistleblowing. In particular, we present a fully ad-hoc solution that allows *anyone* getting to hold a valid ring signature to send a confidential message to the anonymous signer (whistleblower).

Related Work. SPHFs (denoted as hash proof systems) were initially used to construct CCA2 secure public key encryption [CS98] without requiring the random oracle heuristic. Later it was observed that SPHFs are sufficient to construct such encryption schemes [CS02]. They use the SPHF exactly the other way round as we use it, i.e., in their setting decryption is done with the knowledge of the hashing key and without the witness. This paradigm can also be viewed as an implicit construction of publicly evaluable pseudorandom functions [CZ14].

Hybrid Encryption. Kurosawa and Desmedt [KD04] used the paradigm described

[2] Our approach is also easily portable to the SXDH setting (and thus relying on DDH).

above for hybrid encryption. A series of works follow their paradigm (e.g., [KPSY09]) and use SPHFs to obtain CCA2 secure hybrid encryption schemes. Similar to [CS02], they use the SPHF exactly the other way round as we are going to use it.

Key-Exchange. A line of work following Gennaro and Lindell [GL06] uses SPHFs for password-based authenticated key exchange (PAKE) between two parties. This concept was later extended to one-round PAKE [KV11] and generalized to language-authenticated key exchange (LAKE) for various algebraic languages over bilinear groups in [BBC+13a]. We note that follow-up work on various aspects exists. In [BC16] it was very recently shown how to construct so called structure preserving SPHFs which can use GS proofs as witnesses. Even though this is somewhat related to our work it is not useful for us as we want the GS proofs not to be useful to reconstruct the hash value.

2 Preliminaries

Let $x \xleftarrow{R} X$ denote the operation that picks an element x uniformly at random from X. We use $[n]$ to denote the set $\{1, \ldots, n\}$. By $y \leftarrow \mathsf{A}(x)$, we denote that y is assigned the output of the potentially probabilistic algorithm A on input x and fresh random coins and we write $\Pr[\Omega : \mathcal{E}]$ to denote the probability of an event $\mathcal{E}$ over the probability space Ω. A function $\epsilon : \mathbb{N} \to \mathbb{R}^+$ is called negligible if for all $c > 0$ there is a k_0 such that $\epsilon(k) < 1/k^c$ for all $k > k_0$. We use ϵ to denote such a negligible function.

Definition 1 (Bilinear Map). *Let $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ be cyclic groups of prime order p. A bilinear map $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map, where it holds for all $(u, v, a, b) \in \mathbb{G}^2 \times \mathbb{Z}_p^2$ that $e(u^a, v^b) = e(u, v)^{ab}$, and $e(g, g) \neq 1$.*

We typeset $\mathbb{G}_T$ elements in boldface, e.g., $\mathbf{g} = e(g, g)$. Besides the symmetric (Type-1) setting presented above, one can use the asymmetric setting (Type-2 or Type-3). Here, the bilinear map is defined with respect to two different source groups, i.e., $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with $\mathbb{G}_1 \neq \mathbb{G}_2$. In the Type-2 setting an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ exists, whereas such an isomorphism is unknown for the Type-3 setting. Although we have chosen to present our results in the Type-1 setting, it is important to note that our results translate to the asymmetric setting. Such translations can already be nicely automated [AGH15].

Definition 2 (Bilinear Group Generator). *Let* BGGen *be an algorithm which takes a security parameter κ and generates a bilinear group $\mathsf{BG} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ in the Type-1 setting, where the common group order of $\mathbb{G}$ and $\mathbb{G}_T$ is a prime p of bitlength κ, e is a pairing and g is a generator of $\mathbb{G}$.*

Definition 3 (Decision Linear Assumption). *Let $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$. The DLIN assumption states that for all PPT adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that:*

$$\Pr\left[\begin{array}{l} b \xleftarrow{R} \{0,1\},\ g_1, g_2 \xleftarrow{R} \mathbb{G},\ r, s, t \xleftarrow{R} \mathbb{Z}_p, \\ b^* \leftarrow \mathcal{A}\big(\mathsf{BG}, g_1, g_2, g_1^r, g_2^s, g^{b\cdot(r+s)+(1-b)\cdot t}\big) \end{array} : b = b^*\right] \leq 1/2 + \epsilon(\kappa).$$

Universal Hashing. Subsequently, we recall the notion of families of universal hash functions and the leftover hash lemma [HILL99]. We, thereby, align our definitions with [KPSY09] and allow arbitrary domains $\mathcal{X}$ for the hash functions.

Definition 4 (Universal Hash Function Family). *Let $\mathcal{H} = \{\mathsf{H}_y\}_{y\in\{0,1\}^k}$ be a family of hash functions $\mathsf{H}_y : \{0,1\}^k \times \mathcal{X} \to \{0,1\}^\ell$ indexed by a key $y \in \{0,1\}^k$. $\mathcal{H}$ is universal, if for all $x \in \mathcal{X}, x' \in \mathcal{X} \setminus \{x\}$ it holds that*

$$\Pr\left[\mathsf{H}_y \xleftarrow{R} \mathcal{H} : \mathsf{H}_y(x) = \mathsf{H}_y(x')\right] = 2^{-\ell}.$$

Lemma 1 (Leftover Hash Lemma). *Let X be a random variable with support $\mathcal{X}$, let $\delta \geq -\log(\max_{x\in\mathcal{X}} \Pr[X = x])$ and let $\mathcal{H}$ be a family of universal hash functions $\mathsf{H}_y : \{0,1\}^k \times \mathcal{X} \to \{0,1\}^\ell$. Then, for any $\mathsf{H}_y \xleftarrow{R} \mathcal{H}$, we have that $\frac{1}{2}\sum_{z\in\{0,1\}^\ell} \left|\Pr[\mathsf{H}_y(X) = z] - 2^{-\ell}\right| \leq 2^{(\ell-\delta)/2}$.*

Symmetric Encryption. In the following, we recall the definition of symmetric encryption schemes Σ, which we adapt from [KL07]. Analogous to [KD04], we do not explicitly model a key generation algorithm and treat the keys as uniformly random bitstrings $\{0,1\}^{\ell_{\Sigma,\kappa}}$.

Definition 5 (Symmetric Encryption Scheme). *A symmetric encryption scheme Σ is a tuple of PPT algorithms which are defined as follows:*

$\mathsf{Enc}(k, m)$: *Takes a key k and a message m as input and outputs a ciphertext c.*
$\mathsf{Dec}(k, c)$: *Takes a key k and a ciphertext c as input and outputs a message m or $\perp$.*

We require Σ to be correct and to provide ciphertext indistinguishable in the presence of an eavesdropper (IND-EAV; clearly implied by IND-CPA and IND-CCA2). The respective definitions are provided in Appendix A.1.

Groth-Sahai (GS) Non-Interactive Zero-Knowledge Proofs. GS [GS08, GS07] proofs are non-interactive witness-indistinguishable (NIWI) and zero-knowledge (NIZK) proofs for the satisfiability of various types of equations defined over bilinear groups. We require proofs for the satisfiability of pairing product equations (PPEs) in the DLIN setting of the form

$$\prod_{i=1}^{n} e(A_i, \underline{Y_i}) \cdot \prod_{i=1}^{m} e(\underline{X_i}, B_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{X_i}, \underline{Y_j})^{\gamma_{ij}} = t_T, \tag{1}$$

where $(X_1, \ldots X_m) \in \mathbb{G}^m$, $(Y_1, \ldots, Y_n) \in \mathbb{G}^n$ are the secret vectors (to prove knowledge of) and $(A_1, \ldots, A_n) \in \mathbb{G}^n, (B_1, \ldots, B_m) \in \mathbb{G}^m, (\gamma_{ij})_{i\in[m],j\in[n]} \in \mathbb{Z}_p^{n\cdot m}$, and $t_T \in \mathbb{G}_T$ are public constants. To conduct a proof, one commits to the vectors $(X_i)_{i\in[m]}$ and $(Y_i)_{i\in[n]}$, and uses the commitments instead of the actual values in the PPE. Loosely speaking, the proof π is used to "cancel out" the randomness used in the commitments. However, this does not directly work when using the groups $\mathbb{G}$ and $\mathbb{G}_T$, but requires to project the involved elements to the vector spaces $\mathbb{G}^3$ and $\mathbb{G}_T^9$ in the DLIN setting by using the defined projection

maps and to prove the satisfiability of the PPE using the projected elements and corresponding bilinear map $F : \mathbb{G}^3 \times \mathbb{G}^3 \to \mathbb{G}_T^9$.

More precisely, a GS proof for a PPE allows to prove knowledge of a witness $w = ((X_i)_{i\in[m]}, (Y_i)_{i\in[n]})$ such that the PPE, uniquely defined by the statement $x = ((A_i)_{i\in[n]}, (B_i)_{i\in[m]}, (\gamma_{ij})_{i\in[m],j\in[n]}, t_T)$, is satisfied.

Definition 6. *A non-interactive proof system Π is a tuple of PPT algorithms which are defined as follows:*

$\mathsf{BGGen}(1^\kappa)$: *Takes a security parameter κ as input, and outputs a bilinear group description* BG.

$\mathsf{CRSGen}(\mathsf{BG})$: *Takes a bilinear group description* BG *as input, and outputs a common reference string* crs.

$\mathsf{Proof}(\mathsf{BG}, \mathsf{crs}, x, w)$: *Takes a bilinear group description* BG*, a common reference string* crs*, a statement x, and a witness w as input, and outputs a proof π.*

$\mathsf{Verify}(\mathsf{BG}, \mathsf{crs}, x, \pi)$: *Takes a bilinear group description* BG*, a common reference string* crs*, a statement x, and a proof π as input. It outputs a bit $b \in \{0, 1\}$.*

Since we do not explicitly require the security properties here, we omit them and refer the reader to [GS08] at this point.

2.1 Smooth Projective Hashing

A family of smooth projective hash functions (SPHFs) indexed by parameters pp for some family of languages $\{L_{\mathsf{pp},\mathsf{aux}} \subset X_{\mathsf{pp}}\}_{\mathsf{aux}\in\mathsf{A}}$ and associated witness relations $\{R_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$, mapping onto R_{pp} informally works as follows.[3] The hash value can be computed in two ways: (1) Using the hashing key hk one can compute a hash value for every $x \in X_{\mathsf{pp}}$. (2) Using the projection key hp, one can compute a hash value for every $x \in L_{\mathsf{pp},\mathsf{aux}}$. This method, besides x, also requires a witness w such that $R_{\mathsf{pp},\mathsf{aux}}(x, w) = 1$ to compute the hash value. Thereby, both methods yield the same hash value for any $x \in L_{\mathsf{pp},\mathsf{aux}}$. Below we provide a formal definition of SPHFs, where we closely follow [ACP09].

Definition 7. *An* SPHF *is a tuple of the following PPT algorithms:*

$\mathsf{Setup}(1^\kappa)$: *Takes a security parameter κ and outputs the system parameters* pp.

$\mathsf{HashKG}(\mathsf{pp}, \mathsf{aux})$: *Takes the system parameters* pp *and auxiliary information* aux*, and outputs a hashing key* hk.

$\mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, x)$: *Takes a hashing key* hk*, auxiliary information* aux*, and a word x, and outputs a projection key* hp.

$\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, x)$: *Takes a hashing key* hk*, auxiliary information* aux*, and a word x, and outputs a hash value H.*

$\mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, x, w)$: *Takes a projection key* hp*, auxiliary information* aux*, a word x, and a witness w, and outputs a hash value H.*

A secure SPHF is required to be correct, smooth and pseudo-random. Below, we formally define these properties.

[3] Similar to [ACP09] we additionally partition the language L_{pp} with respect to some auxiliary information $\mathsf{aux} \in \mathsf{A}$, where A is determined by pp. Note that without this partitioning, words $x \in L_{\mathsf{pp}}$ additionally contain aux, i.e., so that $x = (\mathsf{aux}, x')$.

Definition 8 (Correctness). *A* SPHF *is correct, if for all* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ *determining* $\{L_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$*, for all* $\mathsf{aux} \in \mathsf{A}$*, for all* $x \in L_{\mathsf{pp},\mathsf{aux}}$*, for all* w *such that* $R_{\mathsf{pp},\mathsf{aux}}(x,w) = 1$*, for all* $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp},\mathsf{aux})$*, and for all* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},\mathsf{aux},x)$*, it holds that* $\mathsf{Hash}(\mathsf{hk},\mathsf{aux},x) = \mathsf{ProjHash}(\mathsf{hp},\mathsf{aux},x,w)$.

Definition 9 (Smoothness). *A* SPHF *is smooth if for all security parameters* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ *determining* $\{L_{\mathsf{pk},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$*, for all* $\mathsf{aux} \in \mathsf{A}$*, for all* $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp},\mathsf{aux})$*, and for all* $x \notin L_{\mathsf{pk},\mathsf{aux}}$ *it holds that:*

$$\{\mathsf{pp},\mathsf{aux},\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},\mathsf{aux},x), H \leftarrow \mathsf{Hash}(\mathsf{hk},\mathsf{aux},x)\} \approx \{\mathsf{pp},\mathsf{aux},\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},\mathsf{aux},x), H \xleftarrow{R} \mathsf{R}_{\mathsf{pp}})\},$$

where $\approx$ *denotes statistical indistinguishability.*

Definition 10 (Pseudorandomness). *A* SPHF *is pseudorandom if for all security parameters* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$ *determining* $\{L_{\mathsf{pk},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$*, for all* $\mathsf{aux} \in \mathsf{A}$*, for all* $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp},\mathsf{aux})$ *it holds that:*

$$\{\mathsf{pp},\mathsf{aux},x \xleftarrow{R} X_{\mathsf{pp}},\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},\mathsf{aux},x), H \leftarrow \mathsf{Hash}(\mathsf{hk},\mathsf{aux},x)\} \approx \{\mathsf{pp},\mathsf{aux},x \xleftarrow{R} X_{\mathsf{pp}},\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk},\mathsf{aux},x), H \xleftarrow{R} \mathsf{R})\},$$

where $\approx$ *denotes computational indistinguishability.*

3 Witness Encryption

WE was initially defined in [GGSW13] and refined by a stronger adaptive soundness notion in [BH13, BH15]. Since it is beneficial regarding practical efficiency and more suitable for the use of WE in the design of cryptographic protocols, we define WE with respect to a setup (similar to [AFP16, Zha16]) and adjust the definitions accordingly.

Definition 11. *A* WE *scheme is a tuple of PPT algorithms defined as follows:*

$\mathsf{Gen}(1^\kappa)$: *Takes a security parameter* κ *and outputs public parameters* pp.
$\mathsf{Enc}(\mathsf{pp},x,m)$: *Takes public parameters* pp*, some word* x *and a message* m *as input and outputs a ciphertext* c.
$\mathsf{Dec}(w,c)$: *Takes a witness* w *and a ciphertext* c *as input and outputs a message* m *or* $\perp$.

We require a WE scheme to be correct and adaptively sound, as defined below.

Definition 12 (Correctness). *A* WE *scheme is correct, if for all* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa)$ *determining* L_{pp}*, for all* m*, for all* $x \in L_{\mathsf{pp}}$*, and for all witnesses* w *such that* $R_{\mathsf{pp}}(x,w) = 1$*, there exists a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\left[\mathsf{Dec}(w,\mathsf{Enc}(\mathsf{pp},x,m)) = m\right] \geq 1 - \epsilon(\kappa).$$

Definition 13 (Soundness). *A* WE *scheme is sound, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that for all* $x \notin L_{\mathsf{pp}}$

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa),\\ (m_0,m_1,\mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp},x),\ b \xleftarrow{R} \{0,1\},\\ c \leftarrow \mathsf{Enc}(\mathsf{pp},x,m_b),\ b^* \leftarrow \mathcal{A}(c,\mathsf{st})\end{array} : \begin{array}{r} b = b^* \wedge\\ |m_0| = |m_1|\end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

Definition 14 (Adaptive Soundness). *A* WE *scheme is adaptively sound, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

$$\Pr\left[\begin{array}{l}\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa),\\ (x, m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}),\ b \xleftarrow{R} \{0,1\},\\ c \leftarrow \mathsf{Enc}(\mathsf{pp}, x, m_b),\ b^* \leftarrow \mathcal{A}(c, \mathsf{st})\end{array} : \begin{array}{r} b = b^* \ \wedge\ x \notin L_{\mathsf{pp}}\\ \wedge\ |m_0| = |m_1|\end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

It is easy to see that adaptive soundness implies soundness. We call a WE scheme secure, if it is correct and adaptively sound.

3.1 Generic Construction of Bit WE from SPHFs

We are now ready to present our generic construction of a WE scheme from any SPHF. We start with a bit encryption WE scheme (cf. Scheme 1), i.e., we assume the message space $\mathcal{M} = \{0,1\}$. For our construction, it turns out that we only need to assume the existence of SPHFs. We achieve this by using an approach similar to the idea of encrypting bits in the GM encryption scheme [GM84]. In

$\mathsf{Gen}(1^\kappa)$: On input of κ, run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^\kappa)$ and return pp.
$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}$, parse x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}, \mathsf{aux})$ and $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$. If $m = 0$, set $C \xleftarrow{R} \mathsf{R}_{\mathsf{pp}}$ and set $C \leftarrow H$ for $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, \mathsf{aux}, x')$ otherwise. Finally, return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$.
$\mathsf{Dec}(w, c)$: On input of w and c, parse c as $(C, (\mathsf{aux}, x'), \mathsf{hp}, \mathsf{pp})$ and compute $H \leftarrow \mathsf{SPHF.ProjHash}(\mathsf{hp}, \mathsf{aux}, x', w)$. Return 1 if $H = C$ and 0 otherwise.

Scheme 1: WE scheme for bits from SPHFs

particular, we use the fact that without knowledge of hk and a witness w for x it is hard to distinguish a hash value from a uniformly random element in the range R_{pp} of the SPHF. Now, if $m = 0$, then the ciphertext is a randomly sampled element from the range R_{pp}, whereas, if $m = 1$, the ciphertext is the correctly computed hash value. Knowledge of a witness w then allows to recompute the hash value using hp (also included in the ciphertext) and consequently to decide whether $m = 0$ or $m = 1$ has been encrypted.

Theorem 1 (proven in Appendix B.1). *If* SPHF *is correct and smooth, then Scheme 1 is secure.*

Remark 1. Note that pseudo-randomness of the SPHF directly states that soundness of the WE scheme also holds for $x \xleftarrow{R} X_{\mathsf{pp}}$ (i.e., so that x is potentially also in the language).

3.2 Extension to Messages of Arbitrary Length

To obtain a WE scheme for arbitrary message lengths we apply a well known paradigm from hybrid encryption to Scheme 1. In Scheme 2 we present a construction that besides an SPHF requires a universal hash function family $\mathcal{H}$ and

a weakly secure symmetric encryption scheme Σ (only requiring IND-EAV security). Our construction works as follows. It uses a universal hash function $\mathsf{H} \in \mathcal{H}$ on the hash value of the SPHF as a randomness extractor to obtain an encryption key for Σ. Note that for the languages we have in mind (group-dependent languages) one could also use alternative extractors such as [CFPZ09]. Furthermore, depending on the chosen randomness extractor, it might be required to choose a larger security parameter for the SPHF to achieve the desired security parameter in the overall scheme. To capture this, we introduce a polynomial $p(\cdot)$ which is determined by the concrete choice of the primitives underlying this construction.

$\mathsf{Gen}(1^\kappa)$: On input of κ, run $\mathsf{pp}' \leftarrow \mathsf{SPHF.Setup}(1^{p(\kappa)})$, where $p(\cdot)$ is a polynomial determined by the concrete instantiation. Fix a family $\mathcal{H}$ of universal hash functions $\mathsf{H} : \mathsf{R}_{\mathsf{pp}} \to \{0,1\}^{\ell_{\Sigma,\kappa}}$. Return $\mathsf{pp} \leftarrow (\mathsf{pp}', \mathcal{H})$.

$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}^*$, parse pp as $(\mathsf{pp}', \mathcal{H})$, and x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}', \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$ and $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, \mathsf{aux}, x')$. Then choose $\mathsf{H} \xleftarrow{R} \mathcal{H}$, compute $k \leftarrow \mathsf{H}(H)$, $C \leftarrow \Sigma.\mathsf{Enc}(k, m)$ and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$.

$\mathsf{Dec}(w, c)$: On input of w and c, parse c as $(C, (\mathsf{aux}, x'), \mathsf{hp}, \mathsf{pp}, \mathsf{H})$, compute $k \leftarrow \mathsf{H}(\mathsf{SPHF.ProjHash}(\mathsf{hp}, \mathsf{aux}, x', w))$, compute and return $m \leftarrow \Sigma.\mathsf{Dec}(k, C)$.

Scheme 2: WE Scheme from SPHFs for messages of arbitrary length

Theorem 2 (proven in Appendix B.2). *If* SPHF *is correct and smooth,* $\mathcal{H}$ *is a family of universal hash functions* $\mathsf{H} : \mathsf{R}_{\mathsf{pp}} \to \{0,1\}^{\ell_{\Sigma,\kappa}}$*, the symmetric encryption scheme* Σ *is correct and* IND-EAV *secure, and* $p(\cdot)$ *is such that* $2^{(\ell_{\Sigma,\kappa}-\alpha)/2}$*, with* $\alpha = -\log(1/|\mathsf{R}_{\mathsf{pp}}|)$*, is negligible in* κ*, then Scheme 2 is secure.*

4 Efficient SPHFs for Algebraic Languages

Recent expressive SPHFs are mostly constructed to be compatible with the universal composability (UC) framework [Can01]. Such constructions (see, e.g., [BBC+13a]) usually build upon SPHFs based on CCA2 secure (labeled) Cramer-Shoup encryption, and, consequently, often trade maximum efficiency for UC security. We do not aim for UC compatibility, as we focus on constructing particularly efficient instantiations of WE. Additionally, we want to achieve compatibility with the GS proof framework, as our goal is to be able to encrypt with respect to a GS proof. Subsequently, we will gradually develop an SPHF in line with these goals. We start with an SPHF being compatible with GS commitments and then extend this SPHF to cover languages for the satisfiability of PPEs.

In Appendix E we present alternative SPHFs, which can be used when GS compatibility is not required.

4.1 SPHF for Linear Groth-Sahai Commitments

Let the language for the SPHF be defined by the commitments used within the GS proof framework, which we exemplify for the DLIN setting. This brings

us one step closer to our final goal, i.e., to be able to encrypt with respect to a statement proven using the GS proof framework. Before we present our construction, we introduce some additional notation. Let $\circ : \mathbb{G}^{1\times n} \times \mathbb{G}^{1\times n} \to \mathbb{G}^{1\times n}$ and $\cdot : \mathbb{Z}_p \times \mathbb{G}^{1\times n} \to \mathbb{G}^{1\times n}$ denote binary operations on row vectors, i.e., $\circ$ denotes entry-wise multiplications, whereas $\cdot$ denotes entry-wise exponentiation.

Linear GS Commitments. We first recall how a linear GS commitment is formed. Let $(U_1, U_2, U_3) \in \mathbb{G}^3 \times \mathbb{G}^3 \times \mathbb{G}^3$ be the commitment parameters for the DLIN setting, which look as follows:

$$U_1 = (g_1, 1, g)\,, U_2 = (1, g_2, g)\,, U_3 = \rho \cdot U_1 \circ \nu \cdot U_2 = \left(g_1^{\rho}, g_2^{\nu}, g^{\rho+\nu}\right),$$

where $\rho, \nu \xleftarrow{R} \mathbb{Z}_p$. If the commitment parameters are set up in this way, one obtains perfectly binding commitments. In contrast, in the perfectly hiding setup we have that $\log_g U_3 \notin \text{span}(\log_g U_1, \log_g U_2)$. The two setups are computationally indistinguishable under DLIN. Thus, we can align our further explanations to the perfectly binding setup and they equally apply to the perfectly hiding case. To commit to a message $M \in \mathbb{G}$ one chooses $r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_p$ and computes

$$C_M = (1, 1, M) \circ r_1 \cdot (g_1, 1, g) \circ r_2 \cdot (1, g_2, g) \circ r_3 \cdot \left(g_1^{\rho}, g_2^{\nu}, g^{\rho+\nu}\right) = \left(g_1^{r_1+\rho r_3}, g_2^{r_2+\nu r_3}, M \cdot g^{r_1+r_2+r_3(\rho+\nu)}\right).$$

Observe that C_M linearly encrypts M with respect to $((r_1 + \rho r_3), (r_2 + \nu r_3))$.

In Scheme 3, we present the SPHF for linear GS commitments, which borrows construction ideas from [GL06].

$\mathsf{Setup}(1^\kappa)$: On input of κ, run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$, choose $(\rho, \nu) \xleftarrow{R} \mathbb{Z}_p^2$, set $\mathsf{pk} \leftarrow (g_1, g_2, g, g_1^{\rho}, g_2^{\nu}, g^{\rho+\nu})$ and return $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{pk})$.
$\mathsf{HashKG}(\mathsf{pp}, \mathsf{aux})$: On input of pp and aux, return $\mathsf{hk} \leftarrow (\mathsf{pp}, \eta, \theta, \zeta) \xleftarrow{R} \mathbb{Z}_p^3$.
$\mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, x)$: On input of hk, auxiliary information $\mathsf{aux} = M \in \mathbb{G}$ and some word $x = C_M \in \mathbb{G}^3$, where $C_M = (g_1^{r_1} g_1^{\rho r_3}, g_2^{r_2} g_2^{\nu r_3}, M \cdot g^{r_1+r_2+r_3(\rho+\nu)})$, compute and return $\mathsf{hp} \leftarrow (\mathsf{pp}, \mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_3) = (g_1^{\eta} g^{\zeta}, g_2^{\theta} g^{\zeta}, (g_1^{\rho})^{\eta} (g_2^{\nu})^{\theta} (g^{\rho+\nu})^{\zeta})$.
$\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, x)$: On input of $\mathsf{hk} = (\mathsf{pp}, \eta, \theta, \zeta)$, $\mathsf{aux} = M \in \mathbb{G}$ and $x = C_M \in \mathbb{G}^3$, where $C_M = (u, v, e)$, compute and return $H \leftarrow u^{\eta} v^{\theta} (e/M)^{\zeta}$.
$\mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, x, w)$: On input of hp, aux, x and $w = (r_1, r_2, r_3)$, compute and return $H \leftarrow \mathsf{hp}_1^{r_1} \cdot \mathsf{hp}_2^{r_2} \cdot \mathsf{hp}_3^{r_3}$.

Scheme 3: SPHF for the language of linear GS commitments

Theorem 3 (proven in Appendix B.3). *If the DLIN assumption holds, then the* SPHF *in Scheme 3 is secure.*

4.2 Extending Supported Languages

Now, to achieve the desired compatibility with statements of the satisfiability of PPEs proven in the GS proof framework, we extend the SPHF for linear

GS commitments from the previous section. We therefore, borrow ideas from [BBC+13a, BBC+13b]. Our framework is compatible with PPEs of the form

$$\prod_{i=1}^{m} e(A_i, \underline{Y_i}) \cdot \prod_{i=m+1}^{o} e(\underline{X_i}, B_i) \cdot \prod_{i=o+1}^{n} \underline{\mathbf{Z}_i}^{\gamma_i} = \mathbf{B}, \tag{2}$$

where X_i, Y_i and $\mathbf{Z}_i$ remain secret and are encrypted using linear encryption. For the ease of presentation, we use the following simplified equation:

$$\prod_{i=1}^{m} e(A_i, \underline{Y_i}) \cdot \prod_{i=m+1}^{n} \underline{\mathbf{Z}_i}^{\gamma_i} = \mathbf{B}. \tag{3}$$

Note that in a Type-1 setting, this simplification does not even influence the expressiveness. We denote the commitments to Y_i and $\mathbf{Z}_i$, respectively, as $C_i = (u_i, v_i, e_i) \in \mathbb{G}^3$ for $1 \leq i \leq m$ and $\mathbf{C}_i = (\mathbf{u}_i, \mathbf{v}_i, \mathbf{e}_i) \in \mathbb{G}_T^3$ for $m < i \leq n$. The language $L_{\mathsf{pp},\mathsf{PPE}}$ contains all tuples $((C_i)_{i\in[m]}, (\mathbf{C}_i)_{m<i\leq n})$ where the committed values satisfy PPE. Membership in $L_{\mathsf{pp},\mathsf{PPE}}$ is witnessed by the randomness used in the commitments. Further, let $\zeta \xleftarrow{R} \mathbb{Z}_p$ and for $i \in [n]$: $\eta_i, \theta_i \xleftarrow{R} \mathbb{Z}_p$, $\mathsf{hk}_i = (\eta_i, \theta_i, \zeta)$ as well as $\mathsf{hp}_i = (\mathsf{hp}_{i1}, \mathsf{hp}_{i2}, \mathsf{hp}_{i3}) = (g_1^{\eta_i} g^{\zeta}, g_2^{\theta_i} g^{\zeta}, (g_1^{\rho})^{\eta_i} (g_2^{\nu})^{\theta_i} (g^{\rho+\nu})^{\zeta})$. Then, $\mathsf{hk} = (\mathsf{pp}, (\mathsf{hk}_i)_{i\in[n]})$, $\mathsf{hp} = (\mathsf{pp}, (\mathsf{hp}_i)_{i\in[n]})$ and we define

$$\begin{aligned} \mathbf{H}_{\mathsf{Hash}} &:= \mathbf{B}^{-\zeta} \cdot \prod_{i=1}^{m} e(A_i, u_i^{\eta_i} v_i^{\theta_i} e_i^{\zeta}) \cdot \prod_{i=m+1}^{n} (\mathbf{u}_i^{\eta_i} \mathbf{v}_i^{\theta_i} \mathbf{e}_i^{\zeta})^{\gamma_i} = \\ \prod_{i=1}^{m} e(A_i, \mathsf{hp}_{i1}^{r_{i1}} \cdot \mathsf{hp}_{i2}^{r_{i2}} \cdot \mathsf{hp}_{i3}^{r_{i3}}) &\cdot \prod_{i=m+1}^{n} e(g^{\gamma_i}, \mathsf{hp}_{i1}^{r_{i1}} \cdot \mathsf{hp}_{i2}^{r_{i2}} \cdot \mathsf{hp}_{i3}^{r_{i3}}) =: \mathbf{H}_{\mathsf{Proj}}. \end{aligned} \tag{4}$$

Lemma 2. *Using the* SPHF *in Scheme 3 as described above yields a secure* SPHF *for any language covered by Equation* (3).

We prove Lemma 2 in Appendix B.4 and note that an extension to statements of the form in Equation (2) can be done analogous to [BBC+13a, BBC+13b].

5 Encrypting With Respect to a Groth-Sahai Proof

Assume that a prover conducts a GS proof π for the satisfiability of some PPE. Such a proof contains commitments to the witness together with some additional group elements used to "cancel out" the randomness in the commitments. Now, given such a proof π, one can encrypt a message with respect to π using our WE instantiated with the SPHF in Equation (4). The witness to decrypt is the randomness which was used in the commitments contained in π, and consequently the entity who produced π can decrypt.

Scheme 4 compactly sketches our approach, where GS refers to the GS proof system and PPE refers to a paring product equation that can be expressed in our SPHF framework from Section 4. We assume that GS.BGGen and GS.CRSGen have already been run and thus the bilinear group description BG as well as

Decryptor		Encryptor
$\pi \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, \mathsf{PPE}, (Y_i)_{i\in[n]}; r)$		
store r	$\xrightarrow{\pi}$	parse π as $(\mathsf{PPE}, (C_i)_{i\in[n]}, \pi_{\mathsf{GS}})$, set $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{crs})$,
extract r_{com} from r	$\xleftarrow{c}$	$c \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, (\mathsf{PPE}, (C_i)_{i\in[n]}), m)$
$m \leftarrow \mathsf{WE.Dec}(r_{\mathsf{com}}, c)$		

Scheme 4: Encryption of a message with respect to a GS proof

the CRS crs are available to both, the encryptor and the decryptor. Again, for simplicity, we use PPEs of the form in Equation (3) without the $\mathbf{Z}_i$ values. We write a GS proof π as a sequence of commitments $(C_i)_{i\in[n]}$, a corresponding PPE and a proof part π_{GS}. Additionally, we make the randomness r used in GS.Proof explicit and assume that one can efficiently derive the randomness r_{com} used in the commitments $(C_i)_{i\in[n]}$ in π from r. Then, words in the language $L_{\mathsf{pp},\mathsf{PPE}}$ in the WE scheme consists of the commitments $(C_i)_{i\in[n]}$ to the unrevealed values $(Y_i)_{i\in[n]}$. Membership in $L_{\mathsf{pp},\mathsf{PPE}}$ is witnessed by r_{com}.

To formally capture the security we would expect when using WE in this context, we introduce the following definition. Informally, we require that breaking soundness remains intractable even if the statement is in fact in $L_{\mathsf{pp},\mathsf{PPE}}$ and a GS proof for this fact is provided.

Definition 15 (Pseudo-Randomness in the Presence of Proofs). *Let* $\mathsf{BG} \leftarrow \mathsf{GS.BGGen}(1^\kappa)$, $\mathsf{crs} \leftarrow \mathsf{GS.CRSGen}(\mathsf{BG})$, *and* PPE *be a pairing product equation with satisfying witnesses* $((Y_{i_j})_{i\in[n]})_{j\in[m]}$. *A* WE *scheme for the language* $L_{(\mathsf{BG},\mathsf{crs}),\mathsf{PPE}}$ *provides pseudo-randomness in the presence of proofs, if for all PPT adversaries* $\mathcal{A}$ *there exists a negligible function* $\epsilon(\cdot)$ *such that for all* $\ell \in [m]$ *it holds that*

$$\Pr\left[\begin{array}{l} \pi = ((C_i)_{i\in[n]}, \mathsf{PPE}, \pi_{\mathsf{GS}}) \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \\ \mathsf{crs}, \mathsf{PPE}, (Y_{i_\ell})_{i\in[n]}),\ b \xleftarrow{R} \{0,1\}, \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{BG}, \mathsf{crs}, \pi), \\ c \leftarrow \mathsf{WE.Enc}((\mathsf{BG}, \mathsf{crs}), (\mathsf{PPE}, (C_i)_{i\in[n]}), m_b), \\ b^* \leftarrow \mathcal{A}(\mathsf{st}, c) \end{array} : \begin{array}{r} b = b^* \ \wedge \\ |m_0| = |m_1| \end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

Theorem 4 (proven in Appendix B.5). *When instantiating Scheme 4 with a* WE *scheme based on the* SPHF *from Equation* (4)*, the DLIN assumption holds, and there is more than one possible witness for the statement, then Scheme 4 provides pseudo-randomness in the presence of proofs.*

How to Reply an Unknown Whistleblower. Now, we come back to our exemplary application. That is, one can use Scheme 4 together with ring signatures to encrypt a message using only information from a given ring signature (specifying an **NP**-language) such that only the anonymous producer of the ring signature, i.e., Edwarda, can decrypt (as *only* she holds the respective witness).

In particular, for every ring signature scheme where the signature includes a proof π of the satisfiability of a PPE that is (1) compatible with our SPHF framework from Section 4.1, and (2) satisfiable by more than one witness, one can use

the technique presented in Scheme 4 to encrypt messages with respect to π. We note that the anonymity of ring signatures most likely already requires that more than one witness for such a PPE exists. In Appendix C.1 we present a generic construction of ring signatures [Gha13], which is compatible with our framework. This construction satisfies the requirements above, which, together with Theorem 4, yields the following corollary.

Corollary 1. *When using Scheme 4 to encrypt with respect to the π_{L_1}-parts of a ring signature from [Gha13], pseudorandomenss in the presence of proofs holds.*

6 Discussion

To underline the practicality of the introduced techniques, we provide a brief performance analysis in Appendix D. Finally, we briefly discuss two other potential applications of the presented methodologies and leave a rigorous investigation as future work.

Ring Encryption. Group encryption [KTY07] is an existing paradigm that can be seen as the encryption analogue to group signatures. In group encryption, a sender can prepare a ciphertext and convince a verifier that it can be decrypted by an anonymous member of some managed group. Thereby, an opening authority can reveal the identity of the group member that is capable of decrypting. Consequently, group encryption involves a dedicated trusted group manager and provides conditional anonymity, i.e, the trusted opening authority can break the anonymity. It is quite straightforward to construct a group encryption variant in the ring setting, i.e., an *ad-hoc* counterpart to group encryption. That is, anyone can encrypt a message such that it is guaranteed that exactly one unknown member of an ad-hoc ring $\mathcal{R}$ is able to decrypt. Our use of WE in Section 5 can be interpreted as a construction of a variant of *ring encryption*. In particular, it allows anyone to encrypt a message with respect to an ad-hoc group (ring), being represented by a ring signature σ with respect to a ring $\mathcal{R}$ (instead of the ring $\mathcal{R}$ itself). Thereby, exactly one member of the ring, i.e., the signer, can decrypt. Furthermore, even the encrypting party does not know who exactly will be able to decrypt. Nevertheless, we can ensure that only the right party is able to decrypt, while nobody is able to reveal the identity of the party that is able to decrypt.

Mutually Anonymous Key-Exchange. Our method to encrypt with respect to a GS proof or in particular with respect to a ring signature could be applied to language-authenticated key exchange (LAKE). That is, two parties that do not want to reveal their identity to each other (but only their membership to potentially distinct and independently chosen ad-hoc rings) can agree on a common encryption key, i.e., by using an SPHF for languages covering ring signatures as demonstrated in Section 5. Note that this goal is in contrast to the goals of private mutual authentication [JL09] or covert mutual authentication [Jar14], which allows two parties belonging to some managed groups to privately authenticate to each other so that external parties cannot obtain any information

about their identities or not even distinguish an instance of the authentication protocol from a random beacon.

Acknowledgements. We thank the anonymous referees from Crypto'16 and Asiacrypt'16 for their valuable comments.

References

[ABP15] Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In *EUROCRYPT*, 2015.

[ACP09] Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth Projective Hashing for Conditionally Extractable Commitments. In *CRYPTO*, 2009.

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline Witness Encryption. In *ACNS*, 2016.

[AGH15] Joseph A. Akinyele, Christina Garman, and Susan Hohenberger. Automating Fast and Secure Translations from Type-I to Type-III Pairing Schemes. In *CCS*, 2015.

[BBC+13a] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In *PKC*, 2013.

[BBC+13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New Techniques for SPHFs and Efficient One-Round PAKE Protocols. In *CRYPTO*, 2013.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *CRYPTO*, 2004.

[BC16] Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In *ASIACRYPT*, 2016.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO*, 2001.

[BH13] Mihir Bellare and Viet Tung Hoang. Adaptive Witness Encryption and Asymmetric Password-based Cryptography. *IACR Cryptology ePrint Archive*, page 704, 2013.

[BH15] Mihir Bellare and Viet Tung Hoang. Adaptive Witness Encryption and Asymmetric Password-Based Cryptography. In *PKC*, 2015.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. *J. Cryptology*, 22(1), 2009.

[Boy08] Xavier Boyen. The uber-assumption family. In *Pairing*, 2008.

[BPV12] Olivier Blazy, David Pointcheval, and Damien Vergnaud. Round-optimal privacy-preserving protocols with smooth projective hash functions. In *TCC*, 2012.

[Can01] Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *FOCS*, 2001.

[CFPZ09] Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal Randomness Extraction from a Diffie-Hellman Element. In *EUROCRYPT*, 2009.

[COR99] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional Oblivious Transfer and Timed-Release Encryption. In *EUROCRYPT*, 1999.

[CS98] Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *CRYPTO*, 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *EUROCRYPT*, 2002.

[CZ14] Yu Chen and Zongyang Zhang. Publicly evaluable pseudorandom functions and their applications. In *SCN*, 2014.

[EG14] Alex Escala and Jens Groth. Fine-tuning groth-sahai proofs. In *PKC*, 2014.

[FNV15] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. *IACR Cryptology ePrint Archive*, page 740, 2015.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *FOCS*, 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the Implausibility of Differing-Inputs Obfuscation and Extractable Witness Encryption with Auxiliary Input. In *CRYPTO*, 2014.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness Encryption and its Applications. In *STOC*, 2013.

[Gha13] Essam Ghadafi. Sub-linear Blind Ring Signatures without Random Oracles. In *IMACC*, 2013.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to Run Turing Machines on Encrypted Data. In *CRYPTO*, 2013.

[GL06] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange[1]. *ACM Trans. Inf. Syst. Secur.*, 9(2), 2006.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness Encryption from Instance Independent Assumptions. In *CRYPTO*, 2014.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2), 1984.

[GS07] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. Cryptology ePrint Archive, Report 2007/155, 2007.

[GS08] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *EUROCRYPT*, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4), 1999.

[Jag15] Tibor Jager. How to Build Time-Lock Encryption. *IACR Cryptology ePrint Archive*, page 478, 2015.

[Jar14] Stanislaw Jarecki. Practical Covert Authentication. In *PKC*, 2014.

[JL09] Stanislaw Jarecki and Xiaomin Liu. Private Mutual Authentication and Conditional Oblivious Transfer. In *CRYPTO*, 2009.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *CRYPTO*, 2004.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC Press, 2007.

[KPSY09] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A New Randomness Extraction Paradigm for Hybrid Encryption. In *EUROCRYPT*, 2009.

[KTY07] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group Encryption. In *ASIACRYPT*, 2007.

[KV11] Jonathan Katz and Vinod Vaikuntanathan. Round-Optimal Password-Based Authenticated Key Exchange. In *TCC*, 2011.
[LKW15] Jia Liu, Saqib A. Kakvi, and Bogdan Warinschi. Extractable witness encryption and timed-release encryption from bitcoin. *IACR Cryptology ePrint Archive*, page 482, 2015.
[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In *ASIACRYPT*, 2001.
[RSW96] Ronald L. Rivest, Adi Shamir, and David A Wagner. Time-lock puzzles and timed-release crypto. Technical report, Massachusetts Institute of Technology, 1996.
[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT*, 2005.
[UW14] Thomas Unterluggauer and Erich Wenger. Efficient Pairings and ECC for Embedded Systems. In *CHES*, 2014.
[Wat05] Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In *EUROCRYPT*, 2005.
[Wee10] Hoeteck Wee. Efficient Chosen-Ciphertext Security via Extractable Hash Proofs. In *CRYPTO*, 2010.
[Zha16] Mark Zhandry. How to Avoid Obfuscation Using Witness PRFs. In *TCC 2016-A*, 2016.

A Definitions and Security Notions

A.1 Security Definitions of Symmetric Encryption Schemes

Definition 16 (Correctness). *Σ is correct, if for all κ, for all $k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}}$ and for all $m \in \{0,1\}^*$ it holds that* $\Pr\left[\mathsf{Dec}(k, \mathsf{Enc}(k,m)) = m\right] = 1$.

Definition 17 (IND-EAV Security). *Σ is* IND-EAV *secure, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l} k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}},\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}(1^\kappa), \\ b \xleftarrow{R} \{0,1\}, c \leftarrow \mathsf{Enc}(k, m_b), \\ b^* \leftarrow \mathcal{A}(c, \mathsf{st}) \end{array} : \begin{array}{r} b = b^* \\ \wedge\ |m_0| = |m_1| \end{array}\right] \leq 1/2 + \epsilon(\kappa),$$

where $|m|$ denotes the length of message m.

A.2 Digital Signatures

For the sake of completeness we formally recall digital signature schemes below.

Definition 18 (Digital Signatures). *A digital signature scheme* Sig *is a triple* (KeyGen, Sign, Verify) *of PPT algorithms:*

$\mathsf{KeyGen}(1^\kappa)$: *The key generation algorithm that takes a security parameter κ as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space $\mathcal{M}$ (we may omit to mention $\mathcal{M}$).*

$\mathsf{Sign}(\mathsf{sk}, m)$: *The (probabilistic) signing algorithm takes a secret key* sk *and a message $m \in \mathcal{M}$ as input and outputs a signature σ.*

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *The deterministic verification algorithm takes a a public key* pk*, a message $m \in \mathcal{M}$ and a signature σ as input and outputs $b \in \{0,1\}$.*

We require correctness and existential unforgeability under chosen message attacks (EUF-CMA security).

B Security Proofs

B.1 Proof of Theorem 1

Proof (Correctness). We analyze the probability that Scheme 1 is not correct, i.e., the probability that if $m = 0$ and $C \xleftarrow{R} \mathsf{R}$ yields a value such that $C = H$. It is easy to see that this only occurs with negligible probability $1/|\mathsf{R}_{\mathsf{pp}}|$. □

Proof (Adaptive Soundness). We use a sequence of games to prove adaptive soundness.

Game 0: The original adaptive soundness game.

Game 1: As Game 0, but we modify the encryption algorithm as follows:

$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}$, parse x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}, \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$, $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, \mathsf{aux}, x')$. Sample $\boxed{C \xleftarrow{R} \mathsf{R}_{\mathsf{pp}}}$ and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$.

Transition - Game 0 → Game 1: By the smoothness of the SPHF, the adversary's view in Game 1 is statistically close to the view in Game 0.

Game 1 is simulated independent of the bit b and distinguishing it from Game 0 would imply a distinguisher for statistically close distributions. □

B.2 Proof of Theorem 2

Correctness is perfect and straightforward to verify, which is why we omit the proof. Adaptive soundness is proven subsequently.

Proof (Adaptive Soundness). We now show that adaptive soundness holds.

Game 0: The original adaptive soundness game.

Game 1: As Game 0, but we modify the encryption algorithm as follows:

$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}^*$, parse pp as $(\mathsf{pp}', \mathcal{H})$, and x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}', \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$. Then choose $\boxed{H \xleftarrow{R} \mathsf{R}_{\mathsf{pp}}}$, choose $\mathsf{H} \xleftarrow{R} \mathcal{H}$, compute $k \leftarrow \mathsf{H}(H)$, $C \leftarrow \Sigma.\mathsf{Enc}(k, m)$ and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$.

Transition - Game 0 → Game 1: By the smoothness of the SPHF, the adversary's view in Game 1 is statistically close to the view in Game 0.

Game 2: As Game 1, but we further modify the encryption algorithm as follows:

$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}^*$, parse pp as $(\mathsf{pp}', \mathcal{H})$, and x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}', \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$. Then choose $H \xleftarrow{R} \mathsf{R}_{\mathsf{pp}}$, choose $\mathsf{H} \xleftarrow{R} \mathcal{H}$, set $\boxed{k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}}}$, compute $C \leftarrow \Sigma.\mathsf{Enc}(k, m)$ and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$.

Transition - Game 1 → Game 2: By Lemma 1, we know that the statistical difference between the adversary's view in Game 1 and Game 2 is bounded by $2^{(\ell_{\Sigma,\kappa}-\alpha)/2}$, with $\alpha = -\log(1/|\mathsf{R}_{\mathsf{pp}}|)$. Thus, there exists a polynomial $p(\cdot)$ such that the adversary's view in Game 1 and Game 2 are statistically close.

Game 3: In Game 2 we are already free to randomly choose the key for the symmetric encryption scheme. Thus, in Game 3, the environment can engage in an $\mathsf{IND\text{-}EAV}$ game with a challenger $\mathcal{C}$. In particular, once the adversary outputs $(x, m_0, m_1, \mathsf{st})$, the environment forwards (m_0, m_1, st) to $\mathcal{C}$ to obtain the challenge ciphertext from $\mathcal{C}$ and use it as C in the simulation of Enc. Once the adversary outputs b^*, the environment forwards it as it's guess to $\mathcal{C}$.

Transition - Game 2 → Game 3: This is only a conceptual change.

The adversary's success probability in Game 3 is bounded by the success probability in the $\mathsf{IND\text{-}EAV}$ game of Σ; a distinguisher between Game 0 and Game 3 would imply a distinguisher for statistically close distributions. □

B.3 Proof of Theorem 3

Proof (Correctness). Let $L_{\mathsf{pp},M}$ be the language of linear GS commitments C_M to messages M and let pp, hk and hp be generated according to the setup in Scheme 3. In particular, we have $C_M = (g_1^{r_1+\rho r_3}, g_2^{r_2+\nu r_3}, M \cdot g^{r_1+r_2+r_3(\rho+\nu)})$ and $w = (r_1, r_2, r_3)$. Let $H_{\mathsf{Proj}} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, M, x, w)$ and $H_{\mathsf{Hash}} \leftarrow \mathsf{Hash}(\mathsf{hk}, M, x)$, then we have

$$\begin{aligned} H_{\mathsf{Hash}} := u^{\eta} \cdot v^{\theta}({}^{e}/_{M})^{\zeta} = \\ g_1^{\eta(r_1+\rho r_3)} \cdot g_2^{\theta(r_2+\nu r_3)} \cdot g^{\zeta(r_1+r_2+r_3(\rho+\nu))} = \\ g_1^{\eta r_1} g^{\zeta r_1} \cdot g_2^{\theta r_2} g^{\zeta r_2} \cdot g_1^{\rho\eta r_3} g_2^{\nu\theta r_3} g^{(\rho+\nu)\zeta r_3} = \\ \mathsf{hp}_1^{r_1} \cdot \mathsf{hp}_2^{r_2} \cdot \mathsf{hp}_3^{r_3} =: H_{\mathsf{Proj}}. \end{aligned}$$

□

Proof (Smoothness). To prove smoothness, we can assume that we have an invalid commitment to some message M. Any such commitment is of the form $(g_1^{r_1+\rho r_3}, g_2^{r_2+\nu r_3}, M \cdot g^{r_4})$, where $r_4 \neq r_1' + r_2' = (r_1 + \rho r_3) + (r_2 + \nu r_3)$ and thus not a word in the language $L_{\mathsf{pp},M}$. With $\mathsf{hp} = (\mathsf{pp}, g_1^{\eta} g^{\zeta}, g_2^{\theta} g^{\zeta}, g_1^{\rho\eta} g_2^{\nu\theta} g^{(\rho+\nu)\zeta})$, the corresponding hash value is then of the form $H = g_1^{\eta(r_1+\rho r_3)} g_2^{\theta(r_2+\nu r_3)} g^{\zeta r_4}$. Taking the discrete logarithms with respect to g yields

$$\begin{aligned} \log_g H_{\mathsf{Hash}} = x_1\eta(r_1 + \rho r_3) + x_2\theta(r_2 + \nu r_3) + \zeta r_4, \\ \log_g \mathsf{hp}_1 = x_1\eta + \zeta, \\ \log_g \mathsf{hp}_2 = x_2\theta + \zeta, \\ \log_g \mathsf{hp}_3 = x_1\rho\eta + x_2\nu\theta + (\rho + \nu)\zeta. \end{aligned}$$

It is easy to see that the only possibility where $\log_g H \in \mathrm{span}(\log_g \mathsf{hp}_1, \log_g \mathsf{hp}_2, \log_g \mathsf{hp}_3)$ is when $r_4 = (r_1 + \rho r_3) + (r_2 + \nu r_3) = r_1' + r_2'$, i.e., when C_M is in fact in $L_{\mathsf{pp},M}$. Conversely, if $C_M \notin L_{\mathsf{pp},M}$ we have that $r_3 \neq r_1' + r_2'$ and the value H looks perfectly random. □

Proof (Pseudo-Randomness). We prove pseudo-randomness using a sequence of hybrid distributions.

Distribution 0: Let D^0 be the distribution sampled according to the pseudo-randomness definition.

Distribution 1: As D^0, but we choose $r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_p^3$ and set $C_M = (g_1^{r_1} g_1^{\rho r_3}, g_2^{r_2} g_2^{\nu r_3}, M' \cdot g^{r_1+r_2+r_3(\rho+\nu)})$ for some $M' \neq M$.

Transition $\mathsf{D}^0 \to \mathsf{D}^1$: We show that a distinguisher $\mathcal{D}^{0\to 1}$ is a DLIN distinguisher using a hybrid sampler, which—depending on the validity of a DLIN instance—either samples from D^0 or D^1. We obtain a DLIN instance $(\mathsf{BG}, g_1, g_2, g_1^r, g_2^s, g^t)$ and let $C_M = (g_1^r g_1^{\rho r_3}, g_2^s g_2^{\nu r_3}, M' \cdot g^t g^{r_3(\rho+\nu)})$ for some $M' \neq M$. Then, if the DLIN instance is invalid we sample from D^0 (C_M is randomly sampled), whereas we sample from D^1 (C_M is never a commitment to M) if it is valid.

In D^1 we have a distribution as in the smoothness game, i.e., the hash value is perfectly random. D^0 and D^1 are computationally indistinguishable, which completes the proof. □

B.4 Proof of Lemma 2

Proof (Correctness). For simplicity, we can without loss of generality assume that $m = 1, n = 2$. Let (r_{11}, r_{12}, r_{13}) and (r_{21}, r_{22}, r_{23}) be the randomness used to compute the GS commitments to Y_1 and $\mathbf{Z}_2$, respectively. Then, (r_{11}, r_{12}, r_{13}) and (r_{21}, r_{22}, r_{23}) represent the witness. The projective hash value obtained using hp is computed as

$$\begin{aligned}\mathbf{H}_{\mathsf{Proj}} \leftarrow \prod_{i=1}^{1} e(A_i, \mathsf{hp}_{i1}^{r_{i1}} \mathsf{hp}_{i2}^{r_{i2}} \mathsf{hp}_{i3}^{r_{i3}}) \cdot \prod_{i=2}^{2} e(g^{\gamma_i}, \mathsf{hp}_{i1}^{r_{i1}} \mathsf{hp}_{i2}^{r_{i2}} \mathsf{hp}_{i3}^{r_{i3}}) = \\ e(A_1, (g_1^{\eta_1} g^{\zeta})^{r_{11}} (g_2^{\theta_1} g^{\zeta})^{r_{12}} ((g_1^{\rho})^{\eta_1} (g_2^{\nu})^{\theta_1} (g^{\rho+\nu})^{\zeta})^{r_{13}}) \cdot \\ e(g^{\gamma_2}, (g_1^{\eta_2} g^{\zeta})^{r_{21}} (g_2^{\theta_2} g^{\zeta})^{r_{22}} ((g_1^{\rho})^{\eta_2} (g_2^{\nu})^{\theta_2} (g^{\rho+\nu})^{\zeta})^{r_{23}}) \ .\end{aligned}$$

Computing the hash value using hk yields:

$$\begin{aligned}\mathbf{H}_{\mathsf{Hash}} \leftarrow \mathbf{B}^{-\zeta} \cdot \prod_{i=1}^{1} e(A_i, u_i^{\eta_i} v_i^{\theta_i} e_i^{\zeta}) \cdot \prod_{i=2}^{2} (\mathbf{u}_i^{\eta_i} \mathbf{v}_i^{\theta_i} \mathbf{e}_i^{\zeta})^{\gamma_i} = \\ \mathbf{B}^{-\zeta} \cdot e(A_1, (g_1^{r_{11}} g_1^{\rho r_{13}})^{\eta_1} (g_2^{r_{12}} g_2^{\nu r_{13}})^{\theta_1} (Y_1 \cdot g^{r_{11}+r_{12}+r_{13}(\rho+\nu)})^{\zeta}) \cdot \\ (e(g, g_1)^{\eta_2(r_{21}+\rho r_{23})} \cdot e(g, g_2)^{\theta_2(r_{22}+\nu r_{23})} \cdot (\mathbf{Z_2} \cdot e(g, g)^{r_{21}+r_{22}+r_{23}(\rho+\nu)})^{\zeta})^{\gamma_2} \overset{(i)}{=} \\ e(A_1, (g_1^{\eta_1} g^{\zeta})^{r_{11}} (g_2^{\theta_1} g^{\zeta})^{r_{12}} ((g_1^{\rho})^{\eta_1} (g_2^{\nu})^{\theta_1} (g^{(\rho+\nu)})^{\zeta})^{r_{13}}) \cdot \\ e(g^{\gamma_2}, (g_1^{\eta_2} g^{\zeta})^{r_{21}} (g_2^{\theta_2} g^{\zeta})^{r_{22}} ((g_1^{\rho})^{\eta_2} (g_2^{\nu})^{\theta_2} (g^{\rho+\nu})^{\zeta})^{r_{23}}) \ .\end{aligned}$$

where for the step (i), we use that $\mathbf{B} = e(A_1, Y_1) \cdot \mathbf{Z}_2^{\gamma_2}$ by definition. □

Smoothness as well as pseudo-randomness follow from the respective properties of the underlying SPHF, as we will discuss subsequently.

Proof (Smoothness). For smoothness, we can without loss of generality assume that one of the n commitments (linear encryptions) contains a value such that the overall PPE is not satisfied. As Y_i and $\mathbf{Z}_i$ cancel out via multiplication by $\mathbf{B}^{-\zeta}$ when plugging in the commitments into the PPE, we know that—by the smoothness of the underlying SPHF—H looks perfectly random. □

Proof (Pseudo-Randomness). We know that smoothness holds by the smoothness of the underlying SPHF. It is easy to see that a distinguisher between the distributions considered in smoothness and pseudo-randomness, would also imply a distinguisher for the same distributions in the underlying SPHF and thus (as already seen in the proof of Lemma 4) a distinguisher for DLIN. □

B.5 Proof of Theorem 4

To prove pseudo-randomness in the presence of proofs we first prove an additional technical lemma.

Lemma 3. *Let DLIN hold, and let Scheme 3' be defined as Scheme 3, with the following modified* Setup *algorithm:*

$\mathsf{Setup}(1^\kappa)$: *On input of* κ*, run* $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$*, choose* $(\rho, \nu, \boxed{\psi}) \xleftarrow{R} \mathbb{Z}_p^3$*, set* $\mathsf{pk} \leftarrow (g_1, g_2, g, g_1^\rho, g_2^\nu, \boxed{g^\psi})$ *and return* $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{pk})$.

Then, pseudorandomness also holds for Scheme 3'.

Proof. We prove pseudo-randomness of Scheme 3' using a sequence of hybrid distributions:

D^0**:** The original pseudo-randomness distribution.
D^1**:** As D^0 but we sample pk as in the setup algorithm of Scheme 3.
Transition $\mathsf{D}^0 \to \mathsf{D}^1$**:** A distinguisher between D^0 and D^1 distinguishes a perfectly binding pk from a perfectly hiding pk, i.e., is a DLIN distinguisher.

For D^1 we know that pseudorandomness holds under DLIN; D^0 and D^1 are computationally indistinguishable. □

Now, we are ready to prove Theorem 4. It, thereby, suffices to show that the hash value looks random (cf. Proof of Theorem 1 and Theorem 2).

Proof. We prove Theorem 4 using a sequence of games.

Game 0: The original pseudo-randomness in the presence of proofs game.
Game 1: As Game 0, but we set the CRS of the GS proof system up to be perfectly hiding, i.e., $\mathsf{pk} \leftarrow (g_1, g_2, g, g_1^\rho, g_2^\nu, \boxed{g^\psi})$ for $(\rho, \nu, \psi) \xleftarrow{R} \mathbb{Z}_p^3$.
Transition Game 0 → Game 1: A distinguisher $\mathcal{D}^{0\to 1}$ contradicts CRS indistinguishability of GS.

Now, we observe that—in the unconditionally hiding setting—honestly sampling the randomness for the commitments as it is done in $\mathsf{GS.Proof}$ is equivalent to sampling the commitments from $\mathbb{G}^3$ uniformly at random (the vectors U_1, U_2, U_3 in the commitment parameters form a basis of $\mathbb{G}^3$). In this setting proofs unconditionally hide the randomness as soon as there are at least two satisfying witnesses for the respective PPE. Thus, the GS proof π does not contain any information about the used randomness (i.e., the witness to decrypt) and what remains is exactly the same distribution as in the pseudorandomness game of Scheme 3' (which holds by Lemma 3). Furthermore, Game 0 and Game 1 are computationally indistinguishable which completes the proof. □

Alternatively, one could also directly prove Theorem 4 in the uber-assumption framework [Boy08] using the soundness setting together with pseudorandomness under the hard subset membership problem. It suffices to show that the hash value is still indistinguishable from random when additionally given a GS proof for the respective statement. In our setting the proof consists of three group elements containing the discrete logarithms $\sum_{i\in[n]} a_i r_{i1}$, $\sum_{i\in[n]} a_i r_{i2}$, and $\sum_{i\in[n]} a_i r_{i3}$, respectively, where $a_i = \log_g A_i$. Since the (η_i, θ_i)-parts are independently chosen for each hk_i the hash value can (independent of the choice of the values a_i) not be represented as a linear combination of these discrete logarithms as soon as $i > 1$. Now, given this linear independence, it is easy to see that this distinguishing task falls into the uber-assumption framework, with $R = S = \langle 1, x_1, x_2, \rho x_1, \nu x_2, \rho+\nu, (a_i)_{i\in[n]}, \sum_{i\in[n]} a_i \cdot r_{i1}, \sum_{i\in[n]} a_i \cdot r_{i2}, \sum_{i\in[n]} a_i \cdot r_{i3}, (x_1(r_{i1} + \rho r_{i3}))_{i\in[n]}, (x_2(r_{i2} + \nu r_{i3}))_{i\in[n]}, (r_{i1} + r_{i2} + r_{i3}(\rho + \nu))_{i\in[n]}, (x_1\eta_i + \xi)_{i\in[n]}, (x_2\theta_i + \xi)_{i\in[n]}, (x_1\rho\eta_i + x_2\nu\theta_i + (\rho+\nu)\xi)_{i\in[n]}\rangle$, $T = \langle 1\rangle$, $f = \langle \sum_{i\in[n]} a_i(\eta_i x_1 (r_{i1} + \rho r_{i3}) + \theta_i x_2(r_{i2} + \nu r_{i3}) + \xi(r_{i1} + r_{i2} + (\rho + \nu)r_{i3}))\rangle$, where all polynomials are in the variables, $(r_{i1}, r_{i2}, r_{i3})_{i\in[n]}$ and $i > 1$.

C Ring Signatures

Ring signature schemes [RST01] are a variant of signature schemes that allow a member of an ad-hoc group $\mathcal{R}$ (the so called ring), defined by the member's public verification keys, to anonymously sign a message on behalf of $\mathcal{R}$. Given a ring signature and all public keys for $\mathcal{R}$, one can verify the validity of such a signature with respect to $\mathcal{R}$, but it is infeasible to identify the actual signer. Subsequently, we formally define ring signature schemes (adopted from [Gha13]).

Definition 19 (Ring Signature Scheme). *A ring signature scheme* RS *is a tuple* $\mathsf{RS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *of PPT algorithms, which are defined as follows:*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes as input a security parameter* κ *and outputs public parameters* pp *(including a description of the message space* $\mathcal{M}$*).*

$\mathsf{KeyGen}(\mathsf{pp})$: *This algorithm takes as input the public parameters* pp *and outputs a keypair* $(\mathsf{sk}, \mathsf{pk})$*.*

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_i, m, \mathcal{R})$: *This algorithm takes as input the public parameters* pp*, a secret key* sk_i*, a message* $m \in \mathcal{M}$ *and a ring* $\mathcal{R} = (\mathsf{pk}_j)_{j\in[n]}$ *of* n *public keys such that* $\mathsf{pk}_i \in \mathcal{R}$*. It outputs a signature* σ*.*

$\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathcal{R})$: *This algorithm takes as input the public parameters* pp*, a message* $m \in \mathcal{M}$*, a signature* σ *and a ring* $\mathcal{R}$*. It outputs a bit* $b \in \{0,1\}$*.*

A secure ring signature scheme is correct, unforgeable, and anonymous. Informally those properties are defined as follows, where we omit the obvious correctness definition. Unforgeability requires that when holding no secret key sk_i that corresponds to a public key $\mathsf{pk}_i \in \mathcal{R}$, one cannot issue valid signatures with respect to arbitrary such rings $\mathcal{R}$. Anonymity requires that it is infeasible to tell which ring member produced a certain signature. For formal definitions we refer the reader to [BKM09].

C.1 Generic Construction of Ring Signatures

We subsequently recall a generic construction of ring signatures for bilinear groups [Gha13]. Ghadafi presents a generic construction (and two possible instantiations) of ring signatures for symmetric and asymmetric prime-order bilinear groups. We recall the generic construction and one instantiation subsequently, where we adapt the notation to ours. Here, Sig denotes an $\mathsf{EUF\text{-}CMA}$ secure digital signature scheme (cf. Appendix A.2) defined over bilinear groups with message space $\mathcal{M}$. That is, it has an additional algorithm $\mathsf{Sig.Setup}(1^\kappa)$ that outputs a bilinear group description BG and all other algorithms take BG as an additional input. Furthermore, GS denotes the non-interactive GS proof system. Henceforth, let the output of $\mathsf{Sig.Setup}$ and $\mathsf{GS.BGGen}$ be compatible, i.e., BG generated by one of these algorithms can be used in both systems. Finally, we can view signatures output by $\mathsf{Sig.Sign}$ as being of the form $\sigma = \{\sigma_j\}_{j\in[n]}$, i.e., they may consists of several elements, and each σ_j that depends on the secret key sk is a group element. We denote the subset of σ that depends on sk as $\overline{\sigma}$ and use $\underline{\sigma}$ for its complement, i.e., $\sigma = \underline{\sigma} \cup \overline{\sigma}$.

Before we introduce the generic ring signature scheme in Scheme 5, we define the **NP**-relations R_1 and R_2 corresponding to the languages L_1 and L_2, respectively. Thereby, $\mathcal{R}$ denotes the ring including a public verification key pk_i so that the signature is produced with the corresponding sk_i. Moreover, $F : \{0,1\}^* \to \mathcal{M}$ is drawn uniformly at random from a family of collision resistant hash functions.

$$((m, \underline{\sigma}, \mathcal{R}), (\mathsf{pk}_i, \overline{\sigma})) \in R_1 \iff \mathsf{Sig.Verify}(\mathsf{pk}_i, F(m||\mathcal{R}), \underline{\sigma} \cup \overline{\sigma}),$$
$$(\mathcal{R}, \mathsf{pk}_i) \in R_2 \iff \mathsf{pk}_i \in \mathcal{R}.$$

Proving knowledge of a witness $(\mathsf{pk}_i, \overline{\sigma})$ for R_1 requires that the verification relation of the underlying signature scheme can be proven using GS. To prove knowledge of a witness pk_i for R_2 using GS, efficient techniques are discussed in [Gha13] and not recalled here. We note that one needs to simultaneously prove knowledge of witnesses for both relations, which can be achieved by reusing the respective GS commitments corresponding to R_1 in R_2 or vice versa (cf. [EG14]).

Theorem 5 ([Gha13]). *Scheme 5 is a secure ring signature scheme, if* Sig *is* $\mathsf{EUF\text{-}CMA}$ *secure,* F *is collision resistant, and* GS *is sound and witness indistinguishable.*

$\mathsf{Setup}(1^\kappa)$: On input of κ, run $\mathsf{BG} \leftarrow \mathsf{Sig.Setup}(1^\kappa)$, $\mathsf{crs} \leftarrow \mathsf{GS.CRSGen}(\mathsf{BG})$, choose a collision resistant hash function $F : \{0,1\}^* \to \mathcal{M}$ and return $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{crs}, F)$.

$\mathsf{KeyGen}(\mathsf{pp})$: On input of pp, parse pp as $(\mathsf{BG}, \mathsf{crs}, F)$, run $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa, \mathsf{BG})$ and return $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_i, m, \mathcal{R})$: On input of pp, sk_i, $m \in \{0,1\}^*$ and $\mathcal{R}$, parse pp as $(\mathsf{BG}, \mathsf{crs}, F)$, run $\sigma \leftarrow \mathsf{Sig.Sign}(\mathsf{BG}, \mathsf{sk}_i, F(m||\mathcal{R}))$, compute $\pi_{L_1} \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, (m, \underline{\sigma}, \mathcal{R}, \mathsf{Sig}), (\mathsf{pk}_i, \overline{\sigma}))$, and $\pi_{L_2} \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, \mathcal{R}, \mathsf{pk}_i)$. In the end, return $\sigma \leftarrow (\underline{\sigma}, \pi_{L_1}, \pi_{L_2})$.

$\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathcal{R})$: On input of pp, m, σ and $\mathcal{R}$, parse pp as $(\mathsf{BG}, \mathsf{crs}, F)$, check whether $\mathsf{GS.Verify}(\mathsf{BG}, \mathsf{crs}, (m, \underline{\sigma}, \mathcal{R}, \mathsf{Sig}), \pi_{L_1}) = 1 \ \wedge \ \mathsf{GS.Verify}(\mathsf{BG}, \mathsf{crs}, \mathcal{R}, \pi_{L_2}) = 1$ and return 1 if so and 0 otherwise.

Scheme 5: Generic construction of ring signatures [Gha13]

Waters Signature Based Ring Signature Instantiation. A very simple scheme, where our generic methodology can be straightforwardly applied, is the instantiation of the generic ring signature construction from [Gha13] based on Waters signatures [Wat05]. See Scheme 6, where we also explicitly present the relation π_{L_1} which is used for the WE scheme. For this construction, Ghadafi shows the following.

$\mathsf{Setup}(1^\kappa)$: On input of κ, run $\mathsf{BG} \leftarrow \mathsf{Sig.Setup}(1^\kappa)$, $\mathsf{crs} \leftarrow \mathsf{GS.CRSGen}(\mathsf{BG})$ and choose a collision resistant hash function $F : \{0,1\}^* \to \{0,1\}^k$. Then, choose $a \xleftarrow{R} \mathbb{Z}_p$, and for all $i \in [k] : U_i \xleftarrow{R} G$. Set $A \leftarrow g^a$, $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{crs}, F, A, (U_i)_{i\in[k]})$.

$\mathsf{KeyGen}(\mathsf{pp})$: On input of pp, parse pp as $(\mathsf{BG}, \mathsf{crs}, F, A, (U_i)_{i\in[k]})$, choose $b \xleftarrow{R} \mathbb{Z}_p$ and compute $B \leftarrow g^b$. In the end, return $(\mathsf{sk}, \mathsf{pk}) \leftarrow ((A^b, B), B)$.

$\mathsf{Sign}(\mathsf{pp}, \mathsf{sk}_i, m, \mathcal{R})$: On input of pp, sk_i, m, $\mathcal{R}$, parse pp as $(\mathsf{BG}, \mathsf{crs}, F, A, (U_i)_{i\in[k]})$, compute $h \leftarrow F(m||\mathcal{R})$ and parse h as $(h_i, \ldots, h_k) \in \{0,1\}^k$, compute $H_W = U_0 \prod_{i=1}^k U_i^{h_i}$. In the end, choose $r \xleftarrow{R} \mathbb{Z}_p$, compute $(\sigma_1, \sigma_2) \leftarrow (g^r, \mathsf{sk}_i[1] \cdot {H_W}^r)$, $\pi_{L_1} \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, (\mathsf{pp}, m, \sigma_1, \mathcal{R}, \mathsf{Sig}), (\mathsf{sk}_i[2], \sigma_2))$ for R_1, $\pi_{L_2} \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, \mathcal{R}, \mathsf{pk}_i)$ for R_2 and return $\sigma \leftarrow (\sigma_1, \pi_1, \pi_2)$.

$\mathsf{Verify}(\mathsf{pp}, m, \sigma, \mathcal{R})$: On input of pp, m, σ and $\mathcal{R}$, parse pp as $(\mathsf{BG}, \mathsf{crs}, F)$, check whether $\mathsf{GS.Verify}(\mathsf{BG}, \mathsf{crs}, (\mathsf{pp}, m, \sigma_1, \mathcal{R}, \mathsf{Sig}), \pi_{L_1}) = 1 \wedge \mathsf{GS.Verify}(\mathsf{BG}, \mathsf{crs}, \mathcal{R}, \pi_{L_2}) = 1$ and return 1 if so and 0 otherwise.

The relation used in π_{L_1} is defined below, where Sig implicitly defines the PPE. π_{L_2} uses the same relation as in the generic construction.

$$((\mathsf{pp}, m, \sigma_1, \mathcal{R}, \mathsf{Sig}), (B, \sigma_2)) \in R_1 \iff e(A^{-1}, B) \cdot e(g, \sigma_2) = e(\sigma_1, U_0 \prod_{i=1}^{k} U_i^{h_i}) \ \wedge \ (h_1, \ldots, h_k) \leftarrow H(m||\mathcal{R}).$$

Scheme 6: Waters signature based ring signature scheme [Gha13]

Theorem 6 ([Gha13]). *The construction in Scheme 6 is secure if the DLIN assumption holds in $\mathbb{G}$, F is a collision resistant hash function, and GS is sound and witness indistinguishable.*

D Efficiency of Our Techniques

We want to emphasize that our techniques are very efficient, and, thus, also appealing from a practical point of view: Counting the expensive operations in $\mathbb{G}$, the SPHF for linear Groth-Sahai commitments in Scheme 3 boils down to 6 exponentiations in ProjKG and 3 exponentiations in Hash and ProjHash, respectively. The operations required when using this SPHF for languages over bilinear groups as demonstrated in Equation (4), are outlined in Table 1. Thereby, m refers to the length of the vector $(Y_i)_{i\in[m]}$, whereas $(m-n)$ refers to the length of the vector $(\mathbf{Z}_i)_{m<i\leq n}$. Here, the computational effort grows linearly in the

Table 1. SPHF for linear GS commitments in PPEs: Expensive operations

	Exp. $\mathbb{G}$	**Exp.** $\mathbb{G}_T$	$e(\cdot,\cdot)$
HashKG	0	0	0
ProjKG	$4n+2$	0	0
Hash	$3m$	$3(n-m)+1$	m
ProjHash	$3n+(n-m)$	0	n

size of the PPE (in particular in n and m, respectively) and is almost as efficient as evaluating the PPE with plain values.

Encrypting w.r.t. Ring Signatures. To underline the practicality of our construction with respect to our exemplary application from Section 5, we analyze the computational effort that is necessary to encrypt a message with respect to a ring signature based on Waters signatures (cf. Scheme 6 in Appendix C.1). Thereby, we assume that the encrypting party exploits synergies from the signature verification of the Waters signature to obtain an even more efficient encryption operation. In particular, we assume that the value $e(\sigma_1, U_0 \prod_{i=1}^{k} U_i^{h_i})$ computed during verification of the Waters signature is cached and is then directly used upon encryption in the computation of the hash value of the SPHF, yielding a PPE with $m = n = 2$ for the underlying SPHF. Then, regarding the expensive operations in $\mathbb{G}$ and $\mathbb{G}_T$, respectively, encryption only requires *16 exponentiations in* $\mathbb{G}$ *and 2 pairings*, and decryption only requires *6 exponentiations in* $\mathbb{G}$ *and 2 pairings*. The operations will most likely be performed on relatively powerful devices such as desktop PCs where the computational overhead will not even be noticeable. Just to give an intuition of how much this will cost on more constrained devices, we assume a portation [AGH15] of our scheme to the Type-3 setting and use performance values of a BN-pairing implementation on an ARM Cortex-M0+ with a drop in hardware accelerator [UW14]. On this platform an exponentiation in $\mathbb{G}_1$ takes 33ms and a pairing takes 164ms, respectively. This means that—even on such a constrained device—encryption can be performed in approximately 1s and decryption in approximately 500ms.

E SPHF for Linear Encryptions

As a basis, we use a DLIN variant [BPV12] of the ElGamal-based SPHF by Gennaro and Lindell [GL06]. Before we continue, we briefly recall linear encryption

as introduced in [BBS04], which is the DLIN equivalent of DDH-based ElGamal encryption.

The setup algorithm chooses a group $\mathbb{G}$ of prime order p generated by g. Key generation amounts to choosing $x_1, x_2 \xleftarrow{R} \mathbb{Z}_p$ and outputting a private key $\mathsf{sk} \leftarrow (x_1, x_2)$ and public key $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow (g^{x_1}, g^{x_2})$. A message $M \in \mathbb{G}$ is encrypted by choosing $r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$ and computing a ciphertext $C_M = (u, v, e) \leftarrow (\mathsf{pk}_1^{r_1}, \mathsf{pk}_2^{r_2}, M \cdot g^{r_1+r_2})$, which in turn can be decrypted by computing $M = e/(u^{1/x_1} \cdot v^{1/x_2})$. It is easy to show (as demonstrated by Boneh et al. in [BBS04]), that the scheme sketched above provides IND-CPA security under the DLIN assumption. It is well known that such a scheme represents a perfectly binding and computationally hiding commitment scheme.

In Scheme 7, we recall the SPHF, where the language $L_{\mathsf{pp},M}$ is with respect to the linear encryption public key pk contained in pp and contains all valid ciphertexts $C_M \in \mathbb{G}^3$. Membership in this language is witnessed by the randomness $r = (r_1, r_2) \in \mathbb{Z}_p^2$ used to compute C_M.

$\mathsf{Setup}(1^\kappa)$: On input of κ, run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$, choose $(x_1, x_2) \xleftarrow{R} \mathbb{Z}_p^2$, set $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) \leftarrow (g^{x_1}, g^{x_2})$, and return $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{pk})$.

$\mathsf{HashKG}(\mathsf{pp}, \mathsf{aux})$: On input of pp and aux, return $\mathsf{hk} \leftarrow (\mathsf{pp}, \eta, \theta, \zeta) \xleftarrow{R} \mathbb{Z}_p^3$.

$\mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, x)$: On input of $\mathsf{hk} = (\mathsf{pp}, \eta, \theta, \zeta)$, $\mathsf{aux} = M \in \mathbb{G}$, and some word $x = C_M \in \mathbb{G}^3$, where $C_M = (\mathsf{pk}_1^{r_1}, \mathsf{pk}_2^{r_2}, M \cdot g^{r_1+r_2})$, compute and return $\mathsf{hp} \leftarrow (\mathsf{pp}, \mathsf{hp}_1, \mathsf{hp}_2) = (\mathsf{pk}_1^{\eta} g^{\zeta}, \mathsf{pk}_2^{\theta} g^{\zeta})$.

$\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, x)$: On input of $\mathsf{hk} = (\mathsf{pp}, \eta, \theta, \zeta)$, $\mathsf{aux} = M \in \mathbb{G}$ and $x = C_M \in \mathbb{G}^3$, where $C_M = (u, v, e)$, compute and return $H \leftarrow u^{\eta} v^{\theta} (e/M)^{\zeta}$.

$\mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, x, w)$: On input of hp, x and $w = (r_1, r_2)$, compute and return $H \leftarrow \mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2}$.

Scheme 7: SPHF for the language of linear ciphertexts

Lemma 4. *If the DLIN assumption holds, then the* SPHF *in Scheme 7 is secure.*

Proof (Correctness). Let $L_{\mathsf{pp},M}$ be the language of linear encryptions of M and let pp, hk and hp be generated according to the setup in Scheme 7. Then $x = C_M = (\mathsf{pk}_1^{r_1}, \mathsf{pk}_2^{r_2}, M \cdot g^{r_1+r_2})$ and $w = (r_1, r_2)$. Let $H_{\mathsf{Proj}} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, M, x, w)$ and $H_{\mathsf{Hash}} \leftarrow \mathsf{Hash}(\mathsf{hk}, M, x)$, then we have

$$H_{\mathsf{Hash}} = u^{\eta} v^{\theta} (e/M)^{\zeta} = \mathsf{pk}_1^{r_1 \eta} \mathsf{pk}_2^{r_2 \theta} g^{(r_1+r_2)\cdot\zeta} = \mathsf{pk}_1^{\eta r_1} g^{\zeta r_1} \mathsf{pk}_2^{\theta r_2} g^{\zeta r_2} = \mathsf{hp}_1^{r_1} \mathsf{hp}_2^{r_2} = H_{\mathsf{Proj}}$$

which proves correctness. □

Proof (Smoothness). To prove smoothness, we can assume that we have an invalid ciphertext to some message M. Any such ciphertext is of the form $(\mathsf{pk}_1^{r_1}, \mathsf{pk}_2^{r_2}, M \cdot g^{r_3})$, where $r_3 \neq r_1 + r_2$ and thus not a word in the language $L_{\mathsf{pp},M}$. With $\mathsf{hp} = (\mathsf{pp}, \mathsf{pk}_1^{\eta} g^{\zeta}, \mathsf{pk}_2^{\theta} g^{\zeta})$, the corresponding hash value is then of

the form $H = \mathsf{pk}_1^{\eta r_1} \mathsf{pk}_2^{\theta r_2} g^{\zeta r_3}$. Taking the discrete logarithms with respect to g yields

$$\begin{aligned} \log_g H &= x_1 \eta r_1 + x_2 \theta r_2 + \zeta r_3, \\ \log_g \mathsf{hp}_1 &= x_1 \eta + \zeta, \\ \log_g \mathsf{hp}_2 &= x_2 \theta + \zeta. \end{aligned}$$

The only possibility where $\log_g H$ can be represented as a linear combination of $\log_g \mathsf{hp}_1$ and $\log_g \mathsf{hp}_2$ is when $r_3 = r_1 + r_2$, i.e., when C_M is in fact in $L_{\mathsf{pp},M}$. Conversely, if $C_M \notin L_{\mathsf{pp},M}$, we have $r_3 \neq r_1 + r_2$ and the value H looks perfectly random. □

Proof (Pseudo-Randomness). We already know that smoothness holds and we now prove pseudo-randomness by showing that a distinguisher between the distributions considered in smoothness and pseudo-randomness is a distinguisher for DLIN. We obtain a DLIN instance $(\mathsf{BG}, g_1, g_2, g_1^r, g_2^s, g^t)$ and sample the ciphertext to M as $(g_1^r, g_2^s, M' \cdot g^t)$ for some $M \neq M'$, set $\mathsf{pk} \leftarrow (g_1, g_2)$, choose $\mathsf{hk} = (\eta, \theta, \zeta) \xleftarrow{R} \mathbb{Z}_p^3$ and set $\mathsf{hp} \leftarrow (g_1^\eta g^\zeta, g_2^\theta g^\zeta)$. Consequently, if we have a valid DLIN instance, we have a distribution as in the smoothness game, whereas we have a distribution as in the pseudo-randomness game if the DLIN instance is random. Assuming the hardness of DLIN contradicts the existence of such an efficient distinguisher. □

E.1 Extending Supported Languages

We can now use the SPHF for linear ElGamal ciphertexts in statements over bilinear groups in a similar way as described for GS commitments in Section 4.1.

Henceforth, let $\zeta \xleftarrow{R} \mathbb{Z}_p$ and for $i \in [n]$: $\eta_i, \theta_i \xleftarrow{R} \mathbb{Z}_p$, $\mathsf{hk}_i = (\eta_i, \theta_i, \zeta)$ as well as $\mathsf{hp}_i = (\mathsf{hp}_{i1}, \mathsf{hp}_{i2}) = (\mathsf{pk}_1^{\eta_i} g^\zeta, \mathsf{pk}_2^{\theta_i} g^\zeta)$. Then, $\mathsf{hk} = (\mathsf{pp}, (\mathsf{hk}_i)_{i \in [n]})$, $\mathsf{hp} = (\mathsf{pp}, (\mathsf{hp}_i)_{i \in [n]})$ and hashing as well as projective hashing are defined as follows.

$$\mathbf{H}_{\mathsf{Hash}} := \mathbf{B}^{-\zeta} \cdot \prod_{i=1}^{m} e(A_i, u_i^{\eta_i} v_i^{\theta_i} e_i^\zeta) \cdot \prod_{i=m+1}^{n} (\mathbf{u}_i^{\eta_i} \mathbf{v}_i^{\theta_i} \mathbf{e}_i^\zeta)^{\gamma_i} =$$

$$\prod_{i=1}^{m} (A_i, \mathsf{hp}_{i1}^{r_{i1}} \mathsf{hp}_{i2}^{r_{i2}}) \cdot \prod_{i=m+1}^{n} e(g^{\gamma_i}, \mathsf{hp}_{i1}^{r_{i1}} \mathsf{hp}_{i2}^{r_{i2}}) =: \mathbf{H}_{\mathsf{Proj}}.$$

Security, of the construction above is easy to verify by the security of Scheme 7 using the strategy in Section 4.2. Thus, we omit the proof and directly state the lemma.

Lemma 5. *Using the* SPHF *in Scheme 7 as described above yields a secure* SPHF *for any language covered by Equation* (3).