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

David Derler[1] and Daniel Slamanig[2]

[1] IAIK, Graz University of Technology, Graz, Austria
david.derler@tugraz.at

[2] AIT Austrian Institute of Technology, Vienna, Austria
daniel.slamanig@ait.ac.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) non-interactive witness-indistinguishable/zero-knowledge 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. To illustrate our techniques, we apply them in context of privacy-preserving exchange of information.

Keywords: Witness encryption, smooth projective hash functions, Groth-Sahai proofs, encryption, privacy.

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 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 $x \in L$

This is a paper to appear in Designs, Codes and Cryptography.

and a ciphertext c hides m if c has been computed with respect to some $x \notin L$.

Constructions of WE for NP. 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.[3] Zhandry in [Zha16] introduced the concept of (extractable) witness PRFs and uses them among others to construct (extractable) WE. Besides, he formalizes *reusable* WE, which introduces an additional global setup and thus allows to reuse certain parameters over many encryptions. This can help to drastically reduce the ciphertext size in WE schemes.

All the above constructions build upon MLMs and/or obfuscation and are thus far from being practical. To this end, Abusalah et al. [AFP16] 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.

WE for Restricted Languages. In concurrent and independent work, Faonio et al. [FNV17] 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]. Earlier work on (private) conditional oblivious transfer [COR99, JL09] can be viewed as an *interactive* variant of (extractable) WE for very specific and restricted languages.[4] Finally, [GGSW13] mentioned along the lines that earlier work on smooth projective hash functions

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

[4] We mention this direction for completeness and stress that we are only interested in non-interactive WE schemes.

(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].

Putting our Work into Context. The approaches in [FNV17, Zha16] bear similarities to how we construct WE. This mainly stems from the conceptual similarity of SPHFs to witness PRFs and extractable hash proof systems, respectively. Essentially, all those primitives have similar goals and thus similar functionality, but they differ regarding the formalization and available instantiations.

With witness PRFs [Zha16], which can be seen as a variant of (extractable) hash proof systems, Zhandry aims at building WE for any **NP**-language and provides an instantiation from MLMs. Consequently, this does not yield a practical instantiation, where our aim is explicitly on constructing practical instantiations of WE and in particular to make them compatible with Groth-Sahai. Moreover, as Zhandry also notes in [Zha16], the security models for witness PRFs and SPHFs are different and not fully compatible, making the approaches incomparable.

The approach to constructing extractable WE from PAoKs (which are instantiated with Ext-HPSs) in [FNV17] is related to our approach, but there are some important differences. Our approach is conceptually simpler, but focuses on WE instead of extractable WE. Given that extractable WE is harder to achieve, WE from SPHFs leads to more expressive languages than extractable WE constructed from Ext-HPSs. For latter, supported languages are very basic and restricted. Using the results from [Wee10] yields instantiations for hard search problems, and in particular factoring and computational Diffie Hellman, but it is not clear if efficient instantiations for more expressive languages can be found. Due to the lack in expressiveness of Ext-HPS as used in [FNV17], their constructions of extractable WE, although practical, are not suitable for supporting languages within Groth-Sahai and we do not see how this could work.

Given the above, it seems that constructing WE from extractable hash proof systems, when targeting constructions for all **NP**-languages either require new assumptions on multilinear maps (as in [Zha16]) yielding impractical instantiations, or yields practical instantiations but for very restricted languages (which are not sufficient for our applications). Consequently, our work addresses a gap which was left open by [FNV17, Zha16] and yields an approach in between these extreme directions. Finally, although Abdalla et al. in [ABP15], informally sketch the construction of WE from SPHFs, this topic has never been formally studied and we are the first to provide a rigorous formal treatment.

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.

1.1 Motivation

While having WE schemes that support *all languages* in **NP** is appealing, it is the main source of inefficiency. Our goal is to study practical instantiations of WE, 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 improve 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 various interesting applications that could benefit from our technique.

1.2 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).[5]

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

– We observe that the existing security notions for WE are unsuited when using WE in combination with other primitives. To this end, we introduce a stronger security notion for WE which considers the combination of WE with GS proofs and prove that our instantiation satisfies this notion.
– 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 discuss two potential applications of our techniques in the context of privacy-preserving exchange of information.

1.3 Related Work

SPHFs (also known as hash proof systems) are hash functions associated to some language L and a hash value can be computed in two ways. Either using the private hashing key (hk) or using the public projection key (hp), whereas latter hashing mode additionally requires a witness w, witnessing the membership of the input to the associated language L. For any $x \in L$ both methods yield the same hash value.

CCA2 secure Encryption. SPHFs were originally used to construct CCA2 secure public key encryption [CS98] that do not require the random oracle heuristic. Later it was observed that SPHFs are sufficient to construct such encryption schemes [CS02]. The elegant idea in [CS98] is to combine ElGamal encryption with a SPHF for the DDH language. The public key includes a projection key of the SPHF and the secret key includes the corresponding hashing key. Roughly, encryption, besides producing a conventional ElGamal ciphertext, computes a projective hash using the randomness of the ElGamal ciphertext as a witness. During decryption, one uses the hashing key to verify whether the hash value has been computed correctly with respect to the ElGamal ciphertext. This paradigm can also be viewed as an implicit construction of publicly evaluable pseudorandom functions [CZ14]. The use of SPHFs in the above approach is exactly the other way round as we are going use it, i.e., in their setting decryption is done with the knowledge of the hashing key and without the witness.

Hybrid Encryption. Kurosawa and Desmedt [KD04] used the paradigm described 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. In particular, they use the hashing key of the SPHF as secret decryption key and the projection key as public encryption key. This setup implicitly defines some language L with an efficiently sampleable witness relation R. Encryption of a message m amounts to randomly sampling $(x, w) \in R$, computing a projective hash value H and using H to extract a key k for a symmetric encryption scheme used to encrypt the message m. To decrypt, one reconstructs

H using the hashing key and the word x, extracts k and uses it for decryption.

Key-Exchange. A line of work following Gennaro and Lindell [GL06] uses SPHFs for password-based authenticated key exchange (PAKE) between two parties. Briefly, the idea is that each party i sends a commitment C_i to the shared password p to the other party. Then, each party i computes a SPHF key pair $(\mathsf{hk}_i, \mathsf{hp}_i)$ for a SPHF defined for a language L, where $(C_i, p) \in R$ if C_i is a commitment to p. Membership in L is witnessed by the randomness r_i used in the commitment C_i. Then, both parties exchange their projection keys hp_i, which allows them to elegantly use the two hashing modes of the SPHF to obtain a shared secret. 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] (and we note that follow-up work on various aspects exists).

Most recently, in [BC16] it was shown how to construct so called structure preserving SPHFs, which can use GS proofs as witnesses. Even though this may sound somewhat related to our work, apart from not constructing WE, the approach in [BC16] to build SPHFs is diametrically opposed to our approach. In particular, our WE approach requires GS proofs to be public and that they must not to be useful to reconstruct the hash value. So, applying our approach to construct WE to the SPHFs in [BC16] does not help us.

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 $\epsilon : \mathbb{N} \rightarrow \mathbb{R}^+$ 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 $e \colon \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ is an efficiently computable map, where it holds for all $(a, b) \in \mathbb{Z}_p^2$ that $e(g^a, g^b) = e(g, g)^{ab}$, and $e(g, g) \neq 1$.*

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., $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ with $\mathbb{G}_1 \neq \mathbb{G}_2$. In the Type-2 setting an efficiently computable isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ 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 $\mathbb{G}$.

To make our notation more succinct, we will henceforth follow [EHK+13] and use $[x]$ to denote $u \in \mathbb{G}$ with $\log_g u = x$. Likewise we will use $[y]_T$ to denote elements $v \in \mathbb{G}_T$ with $\log_{e(g,g)} u = y$. This notation is easily extended to vectors and matrices. We henceforth use additive notation for the groups and may thus simply use linear algebra on the values in the brackets (i.e., the discrete logarithms). Using the bilinear map, we can also make a single multiplication of the discrete logarithms of two elements from $\mathbb{G}$. When we want to denote a pairing product $\sum_{i=1}^{n}[x_i] \cdot [y_i]$ with respect to vectors $[X] = ([x_1], \ldots, [x_n])$ and $[Y] = ([y_1], \ldots, [y_n])$, we write $[X] \bullet [Y]$.

The subsequent assumption is a by now standard assumption introduced in [BBS04].

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:*

$$\Pr\left[\begin{array}{l} b \xleftarrow{R} \{0,1\},\ x_1, x_2 \xleftarrow{R} \mathbb{G},\ r, s, t \xleftarrow{R} \mathbb{Z}_p, \\ b^\star \leftarrow \mathcal{A}\big(\mathsf{BG}, [x_1], [x_2], [x_1 r], [x_2 s], [b(r+s) + (1-b)t]\big) \end{array} \ :\ b = b^\star \right] \leq 1/2 + \epsilon(\kappa).$$

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. We adapt the notion for symmetric encryption schemes Σ from [KL07]. Analogous to [KD04], we do not explicitly model a key generation algorithm and treat the keys as uniformly random bitstrings $k \in \{0,1\}^{\ell_{\Sigma,\kappa}}$.

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

$\mathsf{Enc}(k, \mathsf{m})$: *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). This weak notion is clearly implied by IND-CPA and IND-CCA2 security.

Definition 6 (Correctness). *A symmetric encryption scheme Σ is correct, if for all κ, for all $k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}}$ and for all $\mathsf{m} \in \{0,1\}^*$ it holds that*

$$\Pr\left[\mathsf{Dec}(k, \mathsf{Enc}(k, \mathsf{m})) = \mathsf{m}\right] = 1.$$

Definition 7 (IND-EAV Security). *A symmetric encryption scheme Σ is* IND-EAV *secure, if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that*

$$\Pr\left[\begin{array}{l} k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}},\ (\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}(1^\kappa), \\ b \xleftarrow{R} \{0,1\}, c \leftarrow \mathsf{Enc}(k, \mathsf{m}_b), \\ b^\star \leftarrow \mathcal{A}(c, \mathsf{st}) \end{array} : \begin{array}{r} b = b^\star \ \wedge \\ |\mathsf{m}_0| = |\mathsf{m}_1| \end{array}\right] \leq 1/2 + \epsilon(\kappa),$$

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

Groth-Sahai (GS) Non-Interactive Zero-Knowledge Proofs. Groth-Sahai (GS) proofs [GS08, GS07] 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

$$[A] \bullet [\underline{Y}] + [\underline{X}] \bullet [B] + [\underline{X}] \cdot \Gamma \bullet [\underline{Y}] = [t]_T, \qquad (1)$$

where $[X] \in \mathbb{G}^m$, $[Y] \in \mathbb{G}^n$ are the secret vectors (to prove knowledge of) and $[A] \in \mathbb{G}^n, [B] \in \mathbb{G}^m, \Gamma \in \mathbb{Z}_p^{m \times n}$, and $[t]_T \in \mathbb{G}_T$ are public constants. More generally, underlined elements are secret and all others are public. To conduct a proof, one commits to the vectors X and Y, 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 map the involved elements to the vector spaces $\mathbb{G}^3$ and $\mathbb{G}_T^9$ in the DLIN setting by using the defined maps and to prove the satisfiability of the PPE using the 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 $w = (X, Y)$ such that the PPE, uniquely defined by the statement $x = (A, B, \Gamma, [t]_T)$, is satisfied.

Definition 8 (Non-Interactive Proof System). *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.[6] 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 word x and auxiliary information $\mathsf{aux} \in \mathsf{A}$ where $x \in L_{\mathsf{pp},\mathsf{aux}}$. 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]. For brevity, we henceforth use SPHF to refer to a family of SPHFs indexed by parameters pp.

Definition 9 (Smooth Projective Hash Function). *A* SPHF *for a family of languages* $\{L_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux} \in \mathsf{A}}$ *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.

We require R_{pp} and $L_{\mathsf{pp},\mathsf{aux}}$ to be efficiently sampleable for all pp and aux. A secure SPHF is required to be correct, smooth and pseudo-random. Below, we formally define these properties.

Correctness guarantees that everything works correctly if everyone behaves honestly.

[6] 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 10 (Correctness). *A* SPHF *for a family of languages* $\{L_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$ *is correct, if for all* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, 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* $\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, x) = \mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, x, w)$*.*

Smoothness requires that for any $\mathsf{aux} \in \mathsf{A}$, the hash value looks statistically random for any $x \notin L_{\mathsf{pp},\mathsf{aux}}$.

Definition 11 (Smoothness). *A* SPHF *for a family of languages* $\{L_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$ *is smooth if for all security parameters* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, for all* $\mathsf{aux} \in \mathsf{A}$*, and for all* $x \notin L_{\mathsf{pp},\mathsf{aux}}$ *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 $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp}, \mathsf{aux})$ *and* $\approx$ *denotes statistical indistinguishability.*

Intuitively pseudorandomness requires that for all $\mathsf{aux} \in \mathsf{A}$ the two distributions considered above remain computationally indistinguishable for random $x \in L_{\mathsf{pp},\mathsf{aux}}$. We formally model this using the notion below.

Definition 12 (Pseudorandomness). *A* SPHF *for language* $\{L_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$ *is pseudorandom if for all security parameters* κ*, for all* $\mathsf{aux} \in \mathsf{A}$ *it holds that:*

$$\{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa), \mathsf{aux}, x \xleftarrow{R} L_{\mathsf{pp},\mathsf{aux}}, \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, x), H \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, \\ x)\} \approx \{\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa), \mathsf{aux}, x \xleftarrow{R} L_{\mathsf{pp},\mathsf{aux}}, \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, x), H \xleftarrow{R} \mathsf{R}_{\mathsf{pp}}\},$$

where $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp}, \mathsf{aux})$ *and* $\approx$ *denotes computational indistinguishability.*

Remark 1. It is easy to see that pseudorandomness is implied by smoothness for families of languages where the subset membership problem is hard, i.e., families of languages $\{L_{\mathsf{pp},\mathsf{aux}}\}_{\mathsf{aux}\in\mathsf{A}}$ where it is intractable for any $\mathsf{aux} \in \mathsf{A}$ to distinguish a word in $L_{\mathsf{pp},\mathsf{aux}}$ from a word in $X_{\mathsf{pp}} \setminus L_{\mathsf{pp},\mathsf{aux}}$.

3 Witness Encryption

WE was initially defined in [GGSW13] and refined by a stronger adaptive soundness notion in [BH13, BH15] where the word output by the adversary may depend on the parameters pp. In our context only adaptive soundness is meaningful as we define WE schemes for families of languages indexed by pp, i.e., the language is not fixed before the parameters pp are generated. For brevity, we subsequently simply use "WE scheme" to denote such a scheme. Since it is beneficial regarding practical efficiency and more suitable for the use of WE in the design of cryptographic protocols, we follow [AFP16, Zha16] and define WE with respect to a setup.

Definition 13 (Witness Encryption). *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. *We assume that the public parameters* pp *implicitly define the message space* $\mathcal{M}$.

$\mathsf{Enc}(\mathsf{pp}, x, \mathsf{m})$: *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 14 (Correctness). *A* WE *scheme is correct, if for all* κ, *for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$, *for all* $\mathsf{m} \in \mathcal{M}$, *for all* $x \in L_{\mathsf{pp}}$, *and for all witnesses* w *such that* $R_{\mathsf{pp}}(x, w) = 1$, *it holds that*

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

Definition 15 (Adaptive Soundness). *A* WE *scheme is adaptively sound, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that for all* $x \notin L_{\mathsf{pp}}$ *it holds that*

$$\Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa), \\ (\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{pp}, x),\ b \xleftarrow{R} \{0,1\}, \\ c \leftarrow \mathsf{Enc}(\mathsf{pp}, x, \mathsf{m}_b),\ b^\star \leftarrow \mathcal{A}(c, \mathsf{st}) \end{array} \ : \ \begin{array}{r} b = b^\star\ \wedge \\ |\mathsf{m}_0| = |\mathsf{m}_1| \end{array}\right] \leq 1/2 + \epsilon(\kappa).$$

We call a WE scheme secure if it is correct and adaptively sound.

Remark 2. We note that assuming adaptive soundness of the WE scheme and that the subset-membership problem is hard for domain X_{pp} and language $L_{\mathsf{pp},\mathsf{aux}}$, i.e., the probability of a distinguisher can be bound by $\epsilon(\kappa)$, one can use a standard hybrid argument to show that the probability to break adaptive soundness for a random $x \in L_{\mathsf{pp},\mathsf{aux}}$ is bounded by $\epsilon(\kappa)$.

We, however, stress that adaptive soundness is not sufficient to reach our goals, i.e., to encrypt with respect to a GS proof. To this end, will later define and prove a stronger security notion for our WE construction from SPHFs (cf. Definition 16 and Theorem 4).

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 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 $\mathsf{m} = 0$, then the ciphertext is a randomly sampled element from the range R_{pp}, whereas, if $\mathsf{m} = 1$, 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 $\mathsf{m} = 0$ or $\mathsf{m} = 1$ has been encrypted.

$\underline{\mathsf{Gen}(1^\kappa)}$: Run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^\kappa)$ and return pp.

$\underline{\mathsf{Enc}(\mathsf{pp}, x, \mathsf{m})}$: Parse x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}, \mathsf{aux})$ and $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$. Return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$, where

$$C \xleftarrow{R} \mathsf{R_{pp}} \text{ if } \mathsf{m} = 0, \text{ and } C \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, \mathsf{aux}, x') \text{ otherwise.}$$

$\underline{\mathsf{Dec}(w, c)}$: Parse c as $(C, (\mathsf{aux}, x'), \mathsf{hp}, \mathsf{pp})$ and return 1 if $H = C$ and 0 otherwise, where

$$H \leftarrow \mathsf{SPHF.ProjHash}(\mathsf{hp}, \mathsf{aux}, x', w).$$

Scheme 1: WE scheme for bits from SPHFs

Theorem 1. *If* SPHF *is correct and smooth, then Scheme 1 is secure.*

Proof (Correctness). We analyze the probability that Scheme 1 is not correct, i.e., the probability that if $\mathsf{m} = 0$ 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 $1/|\mathsf{R_{pp}}|$.

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:

$\underline{\mathsf{Enc}(\mathsf{pp}, x, \mathsf{m})}$: Parse x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}, \mathsf{aux})$ and $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$. Return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$, where

$$\boxed{C \xleftarrow{R} \mathsf{R_{pp}}}.$$

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. □

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 a SPHF requires a universal hash function family $\mathcal{H}$ and a symmetric encryption scheme Σ.

Remarkably, our construction only requires the weak notion of IND-EAV security and 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.

$\underline{\mathsf{Gen}(1^\kappa)}$: Run $\mathsf{pp}' \leftarrow \mathsf{SPHF.Setup}(1^{p(\kappa)})$.[a] Fix a family $\mathcal{H}$ of universal hash functions $\mathsf{H} : \mathsf{R}_{\mathsf{pp}} \rightarrow \{0,1\}^{\ell_{\Sigma,\kappa}}$. Return $\mathsf{pp} \leftarrow (\mathsf{pp}', \mathcal{H})$.

$\underline{\mathsf{Enc}(\mathsf{pp}, x, \mathsf{m})}$: Parse pp as $(\mathsf{pp}', \mathcal{H})$, and x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}', \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$, choose $\mathsf{H} \xleftarrow{R} \mathcal{H}$, and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$, where

$$H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, \mathsf{aux}, x'),\ k \leftarrow \mathsf{H}(H),\ C \leftarrow \Sigma.\mathsf{Enc}(k, \mathsf{m}).$$

$\underline{\mathsf{Dec}(w, c)}$: Parse c as $(C, (\mathsf{aux}, x'), \mathsf{hp}, \mathsf{pp}, \mathsf{H})$, obtain $k \leftarrow \mathsf{H}(\mathsf{SPHF.ProjHash}(\mathsf{hp}, \mathsf{aux}, x', w))$, and return m, where

$$\mathsf{m} \leftarrow \Sigma.\mathsf{Dec}(k, C).$$

[a] Note that $p(\cdot)$ is a polynomial determined by the concrete instantiation.

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

Theorem 2. *If* SPHF *is correct and smooth,* $\mathcal{H}$ *is a family of universal hash functions* $\mathsf{H} : \mathsf{R}_{\mathsf{pp}} \rightarrow \{0,1\}^{\ell_{\Sigma,\kappa}}$*, 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.*

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:

$\underline{\mathsf{Enc}(\mathsf{pp}, x, \mathsf{m})}$: Parse pp as $(\mathsf{pp}', \mathcal{H})$, and x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}', \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$, choose $\mathsf{H} \xleftarrow{R} \mathcal{H}$, and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$, where

$$\boxed{H \xleftarrow{R} \mathsf{R}_{\mathsf{pp}}},\ k \leftarrow \mathsf{H}(H),\ C \leftarrow \Sigma.\mathsf{Enc}(k, \mathsf{m}).$$

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:

$\underline{\mathsf{Enc}(\mathsf{pp}, x, \mathsf{m})}$: Parse pp as $(\mathsf{pp}', \mathcal{H})$, and x as (aux, x'), run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp}', \mathsf{aux})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, \mathsf{aux}, x')$, choose $\mathsf{H} \xleftarrow{R} \mathcal{H}$, and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$, where

$$H \xleftarrow{R} \mathsf{R}_{\mathsf{pp}},\ \boxed{k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}}},\ C \leftarrow \Sigma.\mathsf{Enc}(k, \mathsf{m}).$$

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 IND-EAV game with a challenger $\mathcal{C}$. In particular, once the adversary outputs $(x, \mathsf{m}_0, \mathsf{m}_1, \mathsf{st})$, the environment forwards $(\mathsf{m}_0, \mathsf{m}_1, \mathsf{st})$ 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 IND-EAV game of Σ; a distinguisher between Game 0 and Game 3 would imply a distinguisher for statistically close distributions. □

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.

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.

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:

$$[U_1] = ([x_1], [0], [1])\,, \;\; [U_2] = ([0], [x_2], [1])\,,$$
$$[U_3] = \rho \cdot U_1 + \nu \cdot U_2 = ([x_1 \rho], [x_2 \nu], [\rho + \nu])\,,$$

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 $\log_{[1]}[U_3] \notin \mathrm{span}(\log_{[1]}[U_1], \log_{[1]}[U_2])$. 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 $[\mathsf{m}] \in \mathbb{G}$ one chooses $r_1, r_2, r_3 \overset{R}{\leftarrow} \mathbb{Z}_p$ and computes

$$\begin{aligned} C_{[m]} = ([0],[0],[m]) +\ & r_1 \cdot ([x_1],[0],[1]) + \\ & r_2 \cdot ([0],[x_2],[1]) + \\ & r_3 \cdot ([x_1\rho],[x_2\nu],[\rho+\nu]) = \\ & \big([x_1(r_1+\rho r_3)],[x_2(r_2+\nu r_3)],[\mathsf{m}+r_1+r_2+r_3(\rho+\nu)]\big). \end{aligned}$$

Observe that $C_{[\mathsf{m}]}$ linearly encrypts $[\mathsf{m}]$ with respect to randomness $((r_1 + \rho r_3), (r_2 + \nu r_3))$.

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

$\underline{\mathsf{Setup}(1^\kappa)}$: Run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$, choose $(x_1, x_2, \rho, \nu) \overset{R}{\leftarrow} \mathbb{Z}_p^4$, set $\mathsf{pk} \leftarrow ([x_1],[x_2],[1],[x_1\rho],[x_2\nu],[\rho+\nu])$ and return $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{pk})$.

$\underline{\mathsf{HashKG}(\mathsf{pp}, \mathsf{aux})}$: Choose $(\eta, \theta, \zeta) \overset{R}{\leftarrow} \mathbb{Z}_p^3$ and return $\mathsf{hk} \leftarrow (\mathsf{pp}, \eta, \theta, \zeta)$.

$\underline{\mathsf{ProjKG}(\mathsf{hk}, \mathsf{aux}, x)}$: Parse hk as $((\mathsf{BG}, ([x_1],[x_2],[1],[x_1\rho],[x_2\nu],[\rho+\nu])), \eta, \theta, \zeta)$, and return $\mathsf{hp} \leftarrow (\mathsf{pp}, [\mathsf{hp}_1], [\mathsf{hp}_2], [\mathsf{hp}_3])$, where

$$[\mathsf{hp}_1] \leftarrow [x_1\eta] + [\zeta],\ \ [\mathsf{hp}_2] \leftarrow [x_2\theta] + [\zeta],\ \ [\mathsf{hp}_3] \leftarrow [x_1\rho]\cdot\eta + [x_2\nu]\cdot\theta + [\rho+\nu]\cdot\zeta.$$

$\underline{\mathsf{Hash}(\mathsf{hk}, \mathsf{aux}, x)}$: Parse hk as $(\mathsf{pp}, \eta, \theta, \zeta)$, aux as $[\mathsf{m}]$ and x as $([u],[v],[e])$, and return H, where

$$H_{\mathsf{Hash}} \leftarrow [u]\cdot\eta + [v]\cdot\theta + ([e] - [\mathsf{m}])\cdot\zeta.$$

$\underline{\mathsf{ProjHash}(\mathsf{hp}, \mathsf{aux}, x, w)}$: Parse hp as $(\mathsf{pp}, [\mathsf{hp}_1], [\mathsf{hp}_2], [\mathsf{hp}_3])$ and w as (r_1, r_2, r_3), and return H, where

$$H_{\mathsf{Proj}} \leftarrow [\mathsf{hp}_1]\cdot r_1 + [\mathsf{hp}_2]\cdot r_2 + [\mathsf{hp}_3]\cdot r_3.$$

Scheme 3: SPHF for the language of linear GS commitments

Theorem 3. *If the DLIN assumption holds, then the* SPHF *in Scheme 3 is secure.*

Proof (Correctness). Let $L_{\mathsf{pp},[\mathsf{m}]}$ be the language of linear GS commitments $C_{[\mathsf{m}]}$ to messages $[\mathsf{m}]$ and let pp, hk and hp be generated according to the setup in Scheme 3. Furthermore, let $C_{[m]} = ([u],[v],[e]) = ([x_1(r_1+\rho r_3)], [x_2(r_2+\nu r_3)], [\mathsf{m} + r_1 + r_2 + r_3(\rho+\nu)])$, $w = (r_1, r_2, r_3)$, $H_{\mathsf{Proj}} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, [\mathsf{m}], x, w)$, and

$H_{\mathsf{Hash}} \leftarrow \mathsf{Hash}(\mathsf{hk}, [\mathsf{m}], x)$, then we have

$$\begin{aligned} H_{\mathsf{Hash}} := \; & [u] \cdot \eta + [v] \cdot \theta + ([e] - [\mathsf{m}]) \cdot \zeta = \\ & [x_1\eta(r_1 + \rho r_3)] + [x_2\theta(r_2 + \nu r_3)] \cdot [\zeta(r_1 + r_2 + r_3(\rho + \nu))] = \\ & [x_1\eta r_1] + [\zeta r_1] + [x_2\theta r_2] + [\zeta r_2] + [x_1\rho\eta r_3] + [x_2\nu\theta r_3] + [(\rho + \nu)\zeta r_3] = \\ & [\mathsf{hp}_1] \cdot r_1 + [\mathsf{hp}_2] \cdot r_2 + [\mathsf{hp}_3] \cdot r_3 =: H_{\mathsf{Proj}}. \end{aligned}$$

This concludes the proof. □

Proof (Smoothness). To prove smoothness, we can assume that we have an invalid commitment to some message $[\mathsf{m}]$. Any such commitment is of the form $([x_1(r_1 + \rho r_3)], [x_2(r_2 + \nu r_3)], [\mathsf{m} + r_4])$, 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 $L_{\mathsf{pp},[\mathsf{m}]}$. With $\mathsf{hp} = (\mathsf{pp}, [x_1\eta] + [\zeta], [x_2\theta] + [\zeta], [x_1\rho\eta] + [x_2\nu\theta] + [(\rho + \nu)\zeta])$, the corresponding hash value is then of the form $H_{\mathsf{Hash}} = [x_1\eta(r_1 + \rho r_3)] + [x_2\theta(r_2 + \nu r_3)] + [\zeta r_4]$. Taking the discrete logarithms with respect to g yields

$$\begin{aligned} \log_{[1]} H_{\mathsf{Hash}} &= x_1\eta(r_1 + \rho r_3) + x_2\theta(r_2 + \nu r_3) + \zeta r_4, \\ \log_{[1]} [\mathsf{hp}_1] &= x_1\eta + \zeta, \\ \log_{[1]} [\mathsf{hp}_2] &= x_2\theta + \zeta, \\ \log_{[1]} [\mathsf{hp}_3] &= x_1\rho\eta + x_2\nu\theta + (\rho + \nu)\zeta. \end{aligned}$$

It is easy to see that the only possibility where $\log_{[1]} H \in \operatorname{span}(\log_{[1]} [\mathsf{hp}_1], \log_{[1]} [\mathsf{hp}_2], \log_{[1]} [\mathsf{hp}_3])$ is when $r_4 = (r_1 + \rho r_3) + (r_2 + \nu r_3) = r_1' + r_2'$, i.e., when $C_{[\mathsf{m}]}$ is in fact in $L_{\mathsf{pp},[\mathsf{m}]}$. Conversely, if $C_{[\mathsf{m}]} \notin L_{\mathsf{pp},[\mathsf{m}]}$ we have that $r_3 \neq r_1' + r_2'$ and the value H_{Hash} 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 $r_1, r_2, r_3 \overset{R}{\leftarrow} \mathbb{Z}_p^3$ and set $C_{[\mathsf{m}]} = ([x_1 r_1] + [x_1\rho r_3], [x_2 r_2] + [x_2\nu r_3], [\mathsf{m}' + r_1 + r_2 + r_3(\rho + \nu)])$ for some $\mathsf{m}' \neq \mathsf{m}$.

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 $(\mathsf{BG}, [x_1], [x_2], [x_1 r], [x_2 s], [t])$ and let $C_{[\mathsf{m}]} = ([x_1 r] + [x_1\rho r_3], [x_2 s] + [x_2\nu r_3], [\mathsf{m}] + [t] + [r_3(\rho + \nu)])$. Then, if the DLIN instance is valid we sample from D^0, whereas we sample from D^1 if it is invalid.

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. □

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

$$[A] \bullet [\underline{Y}] + [\underline{X}] \bullet [B] + [\underline{Z}]_T \cdot \Gamma = [t]_T, \tag{2}$$

where we have $[A] = ([a_1], \dots [a_n]) \in \mathbb{G}^n$, $[Y] = ([y_1], \dots, [y_n]) \in \mathbb{G}^n$, $[B] = ([b_{n+1}], \dots, [b_{n+m}]) \in \mathbb{G}^m$, $[X] = ([x_{n+1}], \dots, [x_{n+m}]) \in \mathbb{G}^m$, $[Z]_T = ([z_{n+m+1}]_T, \dots, [z_{n+m+o}]_T) \in \mathbb{G}_T^o$, $\Gamma = (\gamma_{n+m+1}, \dots, \gamma_{n+m+o})^\mathsf{T} \in \mathbb{Z}_p^{o\times 1}$. The underlined values in the equation remain secret and are encrypted using linear encryption. For the ease of presentation, we use the following simplified equation (i.e., let $m = 0$):

$$[A] \bullet [\underline{Y}] + [\underline{Z}]_T \cdot \Gamma = [t]_T. \tag{3}$$

Note that in a Type-1 setting, this simplification does not even influence the expressiveness. We denote the commitments to $[y_i]$ and $[z_i]_T$, respectively, as

$$[C_i] = ([u_i], [v_i], [e_i]) \in \mathbb{G}^3 \text{ for } 1 \le i \le n,$$

and

$$[C_i]_T = ([u_i]_T, [v_i]_T, [e_i]_T) \in \mathbb{G}_T^3 \text{ for } n < i \le n + o.$$

Further, let $[C] \in \mathbb{G}^{n\times 3}$ and $[C']_T \in \mathbb{G}_T^{o\times 3}$, where the i-th rows contain the entries of $[C_i]$ and $[C_i]_T$, respectively. Accordingly, we let $R \in \mathbb{Z}_p^{n\times 3}$ and $R' \in \mathbb{Z}_p^{o\times 3}$ be the matrices where the i-th row contains the randomness (r_{i1}, r_{i2}, r_{i3}) being used in commitment $[C_i]$ (resp. $[C_i]_T$). The language $L_{\mathsf{pp},\mathsf{PPE}}$ contains all $([C], [C']_T)$ where the committed values satisfy PPE. Membership in $L_{\mathsf{pp},\mathsf{PPE}}$ is witnessed by (R, R').

Now, for the SPHF, let $\zeta \xleftarrow{R} \mathbb{Z}_p$ and for $i \in [n + o]$: $(\eta_i, \theta_i) \xleftarrow{R} \mathbb{Z}_p^2$, $\mathsf{hk}_i = (\eta_i, \theta_i, \zeta)$ as well as $\mathsf{hp}_i = ([\mathsf{hp}_{i1}], [\mathsf{hp}_{i2}], [\mathsf{hp}_{i3}]) = ([x_1] \cdot \eta_i + [\zeta], [x_2] \cdot \theta_i + [\zeta], [x_1\rho] \cdot \eta_i + [x_2\nu] \cdot \theta_i + [\rho + \nu] \cdot \zeta)$. We use $\mathsf{hk} \in \mathbb{Z}_p^{n\times 3}$ and $\mathsf{hk}' \in \mathbb{Z}_p^{o\times 3}$ to refer to the matrices where the i-th row contains the entries of hk_i (resp. hk_{n+i}). Likewise, we use $[\mathsf{hp}] \in \mathbb{G}^{n\times 3}$ and $[\mathsf{hp}']_T \in \mathbb{G}_T^{o\times 3}$ to refer to the matrices, where the i-th row contains the entries of $[\mathsf{hp}_i]$ (resp. $[\mathsf{hp}_{n+i}]$). Then, we can define

$$\begin{aligned} \mathbf{H}_{\mathsf{Hash}} := [A] \bullet \operatorname{diag}([C] \cdot \mathsf{hk}^\mathsf{T}) + \operatorname{diag}([C']_T \cdot \mathsf{hk}'^\mathsf{T}) \cdot \Gamma - [t]_T \cdot \zeta = \\ [A] \bullet \operatorname{diag}(R \cdot [\mathsf{hp}]^\mathsf{T}) + \operatorname{diag}(R' \cdot [\mathsf{hp}']^\mathsf{T}) \bullet [\Gamma] =: \mathbf{H}_{\mathsf{Proj}}. \end{aligned} \tag{4}$$

Note that the computation of the hash value involves distinct values (η_i, θ_i) in the hashing/projection key per pairing in the pairing products. Intuitively, this is why we require to select elements from the diagonal of $C \cdot \mathsf{hk}^\mathsf{T}$ and $r \cdot \mathsf{hp}^\mathsf{T}$, respectively.

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

Proof (Correctness). For simplicity, we can without loss of generality assume that $n = 1, o = 2$. Then, we have

$$\begin{pmatrix}[C] \\ [C']_T\end{pmatrix} = \begin{pmatrix}[u_1] & [v_1] & [e_1] \\ [u_2]_T & [v_2]_T & [e_2]_T\end{pmatrix}, \begin{pmatrix}R \\ R'\end{pmatrix} = \begin{pmatrix}r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23}\end{pmatrix}, \begin{pmatrix}\mathsf{hk} \\ \mathsf{hk}'\end{pmatrix} = \begin{pmatrix}\eta_1 & \theta_1 & \zeta \\ \eta_2 & \theta_2 & \zeta\end{pmatrix},$$

$$\begin{pmatrix}[\mathsf{hp}] \\ [\mathsf{hp}']\end{pmatrix} = \begin{pmatrix}[x_1\eta_1 + \zeta] & [x_2\theta_1 + \zeta] & [x_1\rho\eta_1 + x_2\nu\theta_1 + (\rho+\nu)\zeta] \\ [x_1\eta_2 + \zeta] & [x_2\theta_2 + \zeta] & [x_1\rho\eta_2 + x_2\nu\theta_2 + (\rho+\nu)\zeta]\end{pmatrix},$$

and the projective hash value obtained using hp is computed as

$$\begin{aligned}\mathbf{H}_{\mathsf{Proj}} \leftarrow\, & [A] \bullet \mathrm{diag}(R \cdot [\mathsf{hp}]^{\mathsf{T}}) + \mathrm{diag}(R' \cdot [\mathsf{hp}']^{\mathsf{T}}) \bullet [\Gamma] \\ & = [a_1] \cdot ([x_1\eta_1 + \zeta] \cdot r_{11} + [x_2\theta_1 + \zeta] \cdot r_{12} \\ & + [x_1\rho\eta_1 + x_2\nu\theta_1 + (\rho+\nu)\zeta] \cdot r_{13}) \\ & + [\gamma_2] \cdot (([x_1\eta_2 + \zeta] \cdot r_{21} + [x_2\theta_2 + \zeta] \cdot r_{22} \\ & + ([x_1\rho\eta_2 + x_2\nu\theta_2 + (\rho+\nu)\zeta)] \cdot r_{23})\ .\end{aligned}$$

Computing the hash value using hk yields:

$$\begin{aligned}\mathbf{H}_{\mathsf{Hash}} \leftarrow\, & [A] \bullet \mathrm{diag}([C] \cdot \mathsf{hk}^{\mathsf{T}}) + \mathrm{diag}([C']_T \cdot \mathsf{hk}'^{\mathsf{T}}) \cdot \Gamma - [t]_T \cdot \zeta \\ & = [a_1] \cdot ([u_1] \cdot \eta_1 + [v_1] \cdot \theta_1 + [e_1] \cdot \zeta) \\ & + ([u_2]_T \cdot \eta_2 + [v_2]_T \cdot \theta_2 + [e_2]_T \cdot \zeta) \cdot \gamma_2 - [t]_T \cdot \zeta \\ & = [a_1] \cdot ([x_1(r_{11} + \rho r_{13})] \cdot \eta_1 + [x_2(r_{12} + \nu r_{13})] \cdot \theta_1 \\ & + ([y_1 + r_{11} + r_{12} + r_{13} \cdot (\rho+\nu)] \cdot \zeta) + ([x_1 \cdot (r_{21} + \rho r_{23})]_T \cdot \eta_2 \\ & + [x_2 \cdot (r_{22} + \nu r_{23})]_T \cdot \theta_2 + [z_2 + r_{21} + r_{22} + r_{23}(\rho+\nu)]_T \cdot \zeta) \cdot \gamma_2 - [t]_T \cdot \zeta \\ & \overset{(i)}{=} [a_1] \cdot ([x_1\eta_1] + [\zeta]) \cdot r_{11} + ([x_2\theta_1] + [\zeta]) \cdot r_{12} \\ & + ([x_1\rho] \cdot \eta_1 + [x_2\nu] \cdot \theta_1 + [\rho+\nu] \cdot \zeta) \cdot r_{13} \\ & + [\gamma_2] \cdot (([x_1\eta_2] + [\zeta]) \cdot r_{21} + ([x_2\theta_2] + [\zeta]) \cdot r_{22} \\ & + ([x_1\rho] \cdot \eta_2 + [x_2\nu] \cdot \theta_2 + [\rho+\nu] \cdot \zeta) \cdot r_{23}).\end{aligned}$$

where for the step (i), we use that $[t]_T = [a_1] \cdot [y_1] + [z_2]_T \cdot \gamma_2$ 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 simplicity, we only consider PPE without the $[z_i]_T$ parts as our argumentation straight forwardly extends to this case (we only consider the discrete logarithms which are the same for the $[z_i]_T$ parts). We can without loss of generality assume that one of the n commitments contains a value such that the overall PPE is not satisfied. Any such commitment is of the form $([x_1(r_{i1} + \rho r_{i3})], [x_2(r_{i2} + \nu r_{i3})], [y_i + r_{i4}])$, where $r_{i4} \neq r'_{i1} + r'_{i2} = (r_{i1} + \rho r_{i3}) + (r_{i2} + \nu r_{i3})$ and thus not a word in the language $L_{\mathsf{pp},\mathsf{PPE}}$. Then, the hash value is defined as:

$$\begin{aligned}\mathbf{H}_{\mathsf{Hash}} &= [A] \bullet [h] - [t]_T \cdot \zeta,\ \text{with } [h] = ([h_1], \ldots, [h_n]),\ \text{and} \\ [h_i] &= [x_1(r_{i1} + \rho r_{i3})] \cdot \eta_i + [x_2(r_{i2} + \nu r_{i3})] \cdot \theta_i + [y_i + r_{i4}] \cdot \zeta.\end{aligned}$$

As the values $[y_i]$ cancel out via subtraction of $[t]_T \cdot \zeta$ when plugging in the commitments into the PPE, it suffices to consider

$$\begin{aligned}\log_{[1]}[h_i] &= x_1\eta_i(r_{i1}+\rho r_{i3}) + x_2\theta_i(r_{i2}+\nu r_{i3}) + r_{i4}\zeta \\ \log_{[1]}[\mathsf{hp}_{i1}] &= x_1\eta_i + \zeta, \\ \log_{[1]}[\mathsf{hp}_{i2}] &= x_2\theta_i + \zeta, \\ \log_{[1]}[\mathsf{hp}_{i3}] &= x_1\rho\eta_i + x_2\nu\theta_i + (\rho+\nu)\zeta.\end{aligned}$$

Now h_i looks perfectly random as it is linearly independent of the projection keys, unless $r_{i4} = (r_{i1}+\rho r_{i3}) + (r_{i2}+\nu r_{i3})$ which only happens if we deal with a valid commitment. This concludes the proof. $\square$

Proof (Pseudo-Randomness). We know that smoothness holds. Using the same argumentation as before, this also implies pseudo-randomness. $\square$

Finally, we note that an extension to statements of the form in Equation (2) can be done analogous to [BBC+13a, BBC+13b].

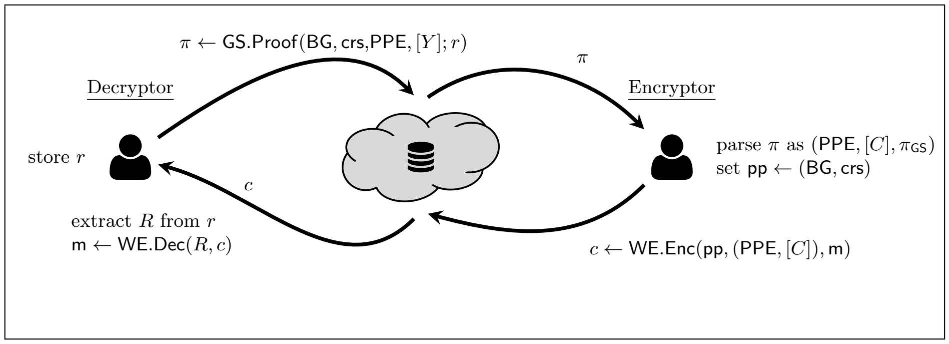
5 Encrypting With Respect to a Groth-Sahai Proof

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 described 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 $\mathsf{GS.BGGen}$ and $\mathsf{GS.CRSGen}$ have already been run and thus the bilinear group description BG as well as the CRS crs are available to both, the encryptor and the decryptor. Again, we use PPEs of the form in Equation (3) without the $[z_i]_T$ values.

We write a GS proof π as a matrix of commitments $[C]$, a corresponding PPE and a proof part π_{GS}. Additionally, we make the randomness r used in $\mathsf{GS.Proof}$ explicit and assume that one can efficiently derive the randomness matrix R corresponding to $[C]$ in π from r. Then, words in the language $L_{\mathsf{pp},\mathsf{PPE}}$ in the WE scheme consist of the commitments $[C]$ to the unrevealed values $[Y]$. Membership in $L_{\mathsf{pp},\mathsf{PPE}}$ is witnessed by R.

Remark 3. One might be inclined to think that it would be sufficient to take just a single GS commitment from a proof π together with the SPHF from Scheme 2. However, then the encryptor would be required to know the value of the $[y_i]$ corresponding to the commitment, i.e., a part of the witness used in the GS proof, which is contrary to using GS in the first place. In contrast, for the solution we propose the knowledge of $[t]_T$ is sufficient.


$\pi \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, \mathsf{PPE}, [Y]; r)$
π
Decryptor
Encryptor
store r
parse π as $(\mathsf{PPE}, [C], \pi_{\mathsf{GS}})$
set $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{crs})$
c
extract R from r
$\mathsf{m} \leftarrow \mathsf{WE.Dec}(R, c)$
$c \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, (\mathsf{PPE}, [C]), \mathsf{m})$

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

To formally capture the security one would expect when using WE in this context, we introduce the following definition. Informally, we require that breaking adaptive soundness of the WE scheme remains intractable even if the statement is in fact in $L_{\mathsf{pp},\mathsf{PPE}}$ and a GS proof for this fact is provided. Since we want the notion to be useful when combining the primitive with other primitives in protocol design, we ask for a rather strong notion—even stronger than the one given in the intuition above. That is, we do not want to assume anything on the distribution of the values defining the PPE and the corresponding witness, i.e., $([A], [t]_T)$ and $[Y]$. We model this by allowing the adversary to adaptively choose the pairing product equation PPE as well as a witness satisfying it. The only part where the adversary is not involved is the computation of the GS proof.

Definition 16 (Pseudo-Randomness in the Presence of Proofs). *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 it holds that*

$$\Pr\left[\begin{array}{l}\mathsf{BG} \leftarrow \mathsf{GS.BGGen}(1^\kappa),\ \mathsf{crs} \leftarrow \mathsf{GS.CRSGen}(\mathsf{BG}),\\ b \xleftarrow{R} \{0,1\},\ (\mathsf{PPE}, [Y], \mathsf{st}) \leftarrow \mathcal{A}(\mathsf{crs}),\\ \pi = ([C], \pi_{\mathsf{GS}}) \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, \mathsf{PPE}, [Y]),\\ c \leftarrow \mathsf{WE.Enc}((\mathsf{BG}, \mathsf{crs}), (\mathsf{PPE}, [C]), b),\\ b^\star \leftarrow \mathcal{A}(\mathsf{st}, c)\end{array} : b = b^\star\right] \leq 1/2 + \epsilon(\kappa),$$

where PPE *is of the form* $([A], [t]_T) = (([a_1], \ldots, [a_n]), [t]_T)$*,* $a_i \neq 0$ *for all* $i \in [n]$*, and* $n > 1$*.*

Theorem 4. *When instantiating Scheme 1 with the* SPHF *from Equation* (4)*, then Scheme 4 provides pseudo-randomness in the presence of proofs in the generic group model.*

Proof. Let $\mathcal{A}$ be a generic PPT adversary, who can obtain random strings as handles for group elements, and can execute group operations by querying those random strings to group operation oracles. W.l.o.g. we let the encodings of elements from $\mathbb{G}$ to be in $\{0,1\}^a$ and the encodings of elements from $\mathbb{G}_T$ to be in $\{0,1\}^b$ with $a < b$ and a, b being such that $\mathcal{A}$ can not efficiently guess values chosen uniformly random from $\{0,1\}^a$ (resp. $\{0,1\}^b$).

We start our simulation by randomly choosing $(\alpha_1, \ldots, \alpha_6) \xleftarrow{R} (\{0,1\}^a)^6$, assigning the polynomials $1, x_1, x_2, x_1\rho, x_2\nu, \rho + \nu$ to these values, and storing $(\alpha_1, 1), (\alpha_2, x_1), (\alpha_3, x_2), (\alpha_4, x_1\rho), (\alpha_5, x_2\nu), (\alpha_6, \rho+\nu)$ in a newly initialized list $L_{\mathbb{G}}$. Furthermore, we initialize an empty list $L_{\mathbb{G}_T}$ to store encoding-tuples for $\mathbb{G}_T$. Then we start $\mathcal{A}$ on $(\alpha_1, \ldots, \alpha_6)$ and simulate the group operation oracles as follows:

Group operation in $\mathbb{G}$*:* Given two encodings α_0, α_1, obtain the corresponding polynomials f_0, f_1 from $L_{\mathbb{G}}$ and return $\perp$ if the lookup fails. Otherwise check whether there is a tuple $(\alpha_\ell, f_0 + f_1)$ in $L_{\mathbb{G}}$, If not, choose $\alpha_\ell \xleftarrow{R} \{0,1\}^a$ and store $(\alpha_\ell, f_0 + f_1)$ in $L_{\mathbb{G}}$. In the end, return α_ℓ.

Inversion in $\mathbb{G}$*:* Given an encoding α, obtain the corresponding polynomials f from $L_{\mathbb{G}}$ and return $\perp$ if the lookup fails. Otherwise check whether there is a tuple $(\alpha_\ell, -f)$ in $L_{\mathbb{G}}$. If not, choose $\alpha_\ell \xleftarrow{R} \{0,1\}^a$, store $(\alpha_\ell, -f)$ in $L_{\mathbb{G}}$. In the end, return α_ℓ.

Pairing: Given two encodings α_0, α_1, obtain the corresponding polynomials f_0, f_1 from $L_{\mathbb{G}}$ and return $\perp$ if the lookup fails. Otherwise check whether there is a tuple $(\alpha_\ell, f_0 \cdot f_1)$ in $L_{\mathbb{G}_T}$, If not, choose $\alpha_\ell \xleftarrow{R} \{0,1\}^b$ and store $(\alpha_\ell, f_0 \cdot f_1)$ in $L_{\mathbb{G}_T}$. In the end, return α_ℓ.

The group operation and inversion oracles for the group $\mathbb{G}_T$ are simulated analogously to the ones for $\mathbb{G}$, with the difference that one samples the encodings from $\{0,1\}^b$ and uses the list $L_{\mathbb{G}_T}$.

If the adversary eventually outputs $(((a_1, \ldots, a_n), t), (y_1, \ldots, y_n), \mathsf{st}) \in (((\{0,1\}^a)^n \times \{0,1\}^b) \times (\{0,1\}^a)^n \times \mathsf{ST})$ representing $(\mathsf{PPE}, [Y], \mathsf{st})$, we abort if one of the encodings has no corresponding polynomial in the respective lists. Otherwise, we define the polynomials $\sum_{i\in[n]} a_i \cdot r_{i1}, \sum_{i\in[n]} a_i \cdot r_{i2}, \sum_{i\in[n]} a_i \cdot r_{i3}, (x_1(r_{i1} + \rho r_{i3}))_{i\in[n]}, (x_2(r_{i2} + \nu r_{i3}))_{i\in[n]}, (r_{i1} + r_{i2} + r_{i3}(\rho+\nu))_{i\in[n]}, (x_1\eta_i + \zeta)_{i\in[n]}, (x_2\theta_i + \zeta)_{i\in[n]}, (x_1\rho\eta_i + x_2\nu\theta_i + (\rho+\nu)\zeta)_{i\in[n]}$ choose an encoding value for each of the polynomials uniformly at random from $\{0,1\}^a$ and insert the respective tuples in $L_{\mathbb{G}}$. Likewise, we define the polynomial $b \cdot \sum_{i\in[n]} a_i(\eta_i x_1(r_{i1} + \rho r_{i3}) + \theta_i x_2(r_{i2} + \nu r_{i3}) + \zeta(r_{i1} + r_{i2} + (\rho+\nu) r_{i3})) + (1-b) \cdot \varXi$, choose an encoding value uniformly random in $\{0,1\}^b$ and insert the respective tuple in $L_{\mathbb{G}_T}$. Note that we slightly abuse the notation and use a_i instead of the actual polynomials assigned to a_i, since our argumentation below holds irrespective of the form of the polynomials assigned to a_i. Also note that we do not need to check whether we have a collision upon insertion in the lists, as every newly introduced polynomial contains at least one new variable. Finally, we start $\mathcal{A}$ on all those encodings and st which was previously output by $\mathcal{A}$, and allow $\mathcal{A}$ to query the group operation oracles again.

Eventually $\mathcal{A}$ outputs its guess $b^\star$ and we choose concrete values for each of the variables $x_1, x_2, \rho, \nu, (r_{i1}, r_{i2}, r_{i3})_{i\in[n]}, (\eta_i, \theta_i)_{i\in[n]}, \zeta, \varXi, b$. If there is no collision in the polynomials in $L_{\mathbb{G}}$ and $L_{\mathbb{G}_T}$, no information about b was leaked and the adversary can only win with probability $1/2$. If there is a collision we have two possibilities: (1) the collision was produced by $\mathcal{A}$ itself, or (2) the collision occurs due to the assignment of concrete values. We subsequently show that neither (1) nor (2) can happen with noticeable probability:

(1) In this case we can take a shortcut: we observe that without using the GS proof, i.e., without using the polynomials $\sum_{i\in[n]} a_i \cdot r_{i1}, \sum_{i\in[n]} a_i \cdot r_{i2}, \sum_{i\in[n]} a_i \cdot r_{i3}$, the adversary cannot produce any collisions as this would imply a generic adversary against pseudo-randomness (which holds by Lemma 2). Thus, we know that any collision must involve polynomials corresponding to the GS proof. In this context, we however observe that—under the constraint that all values $a_i \neq 0$ and $n > 1$—any combination of the polynomials $\sum_{i\in[n]} a_i \cdot r_{i1}, \sum_{i\in[n]} a_i \cdot r_{i2}, \sum_{i\in[n]} a_i \cdot r_{i3}$ with the other available polynomials either yields at least one term containing $r_{ij} \cdot r_{k\ell}$ with $i \neq k \ \vee \ j \neq \ell$ or at least one term containing $r_{i1} \cdot \eta_j$, $r_{i1} \cdot \theta_j$, $r_{i2} \cdot \eta_j$, $r_{i2} \cdot \theta_j$, $r_{i3} \cdot \eta_j$, or $r_{i3} \cdot \theta_j$ with $i \neq j$. From the fact, that the adversary can not increase the degree of the polynomials in $L_{\mathbb{G}_T}$ and in particular not the degree of the polynomial which depends on b, we can conclude that the adversary can not produce a collision on its own.

(2) What remains is to exclude a collision occurring due to the random assignment of values. This case occurs exactly when the difference of the evaluations of two polynomials is equal to 0. Assuming that $\mathcal{A}$ makes q queries, we have $\mathcal{O}(\binom{q}{2})$ possible difference polynomials and—by the Schwartz-Zippel Lemma—the probability for a collision is $\mathcal{O}(q^2/p)$. This concludes the proof. $\square$

As a corollary of the proof above we obtain the following for arbitrary-length messages.

Corollary 1. *When instantiating Scheme 2 with the* SPHF *from Equation* (4)*,* $\mathcal{H}$ *is a family of universal hash functions* $\mathsf{H} : \mathsf{R}_{\mathsf{pp}} \rightarrow \{0,1\}^{\ell_{\Sigma,\kappa}}$*, the symmetric encryption scheme* Σ *is* $\mathsf{IND\text{-}EAV}$ *secure, and* $p(\cdot)$ *is such that* $2^{(\ell_{\Sigma,\kappa}-\alpha)/2}$*, then Scheme 4 provides pseudo-randomness in the presence of proofs in the generic group model.*

6 Discussion and Applications

Subsequently, we want to briefly demonstrate 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 $\mathsf{ProjHash}$, respectively. The operations required when using this SPHF for languages over bilinear groups as demonstrated in Equation (4), are outlined in Table 1. Thereby, n refers to the length of the vector $[Y]$, whereas o refers to the length of the vector $[Z]_T$. Here, the computational effort grows linearly in the size of the PPE (in particular in n and o, respectively) and is almost as efficient as evaluating the PPE with plain values.

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	$4(n+o)+1$	0	0
Hash	$3n$	$3o+1$	n
ProjHash	$3(n+o)+o$	0	$n+o$

Regarding applications, our framework is applicable to extend various Groth-Sahai based privacy-enhancing protocols with encryption features in an ad-hoc fashion. Below, we take a closer look at two potential applications.

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. Using our techniques, 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. In particular, it allows anyone to encrypt a message with respect to $\mathcal{R}$ being represented by a proof of membership of a certain entity in $\mathcal{R}$. Thereby, exactly one member of the ring, i.e., the prover, 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.

As an illustrative example of ring encryption let us assume that a whistleblower wants to leak a secret to some journalist. Therefore, she needs to establish a secure channel to transmit the secret. Clearly, the journalist might not want to publicly reveal that he is willing to publish critical information leaked by a whistleblower. Using our techniques, the journalist can prove membership in some group of trusted journalists (without revealing his identity) so that the whistleblower can use this proof to encrypt the secret in a way that she has a high level of confidence that only a member of the group will be able to read the secret.

Policy-Based Encryption. One could even generalize ring encryption to encryption with respect to arbitrary policies. That is, in the fashion of policy-based signatures [BF14], a proof that a certain policy is satisfied could be used to encrypt.

Mutually Anonymous Key-Exchange. Our method to encrypt with respect to a GS proof 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 groups) can agree on a common encryption key. 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.

7 Conclusion

In this paper we have studied practical WE schemes and focused on schemes that do not cover all languages in **NP** but algebraic languages defined over bilinear groups. We formally showed that such WE constructions can be efficiently instantiated from SPHFs. Regarding the usage of WE in more complex protocols, we observed that the conventional adaptive soundness notion provided by WE schemes is not useful. To this end we introduced the notion of *pseudo-randomness in the presence of proofs*, which makes WE constructions attractive for the usage in more complex scenarios. As a first step towards achieving such practically useful WE schemes, we proved that our constructions satisfy this more realistic notion. Due to the strength of this notion we had to resort to a proof in the generic group model. We see coming up with a proof under a weaker assumption as a major open question. Regarding applications of WE schemes providing this strong notion, we have discussed use cases in the context of privacy preserving exchange of data. Yet, it seems that the approach is a valuable tool for various other practical applications.

Acknowledgements. The authors have been supported by EU H2020 project PRISMACLOUD, grant agreement n°644962. We thank various anonymous referees for their valuable comments.

References

ABP15. Michel Abdalla, Fabrice Benhamouda, and David Pointcheval. Disjunctions for hash proof systems: New constructions and applications. In *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, pages 69–100, 2015.

ACP09. Michel Abdalla, Céline Chevalier, and David Pointcheval. Smooth Projective Hashing for Conditionally Extractable Commitments. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 671–689, 2009.

AFP16. Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline witness encryption. In *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, pages 285–303, 2016.

AGH15. Joseph A. Akinyele, Christina Garman, and Susan Hohenberger. Automating Fast and Secure Translations from Type-I to Type-III Pairing Schemes. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1370–1381, 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*, volume 7778 of *LNCS*, pages 272–291. Springer, 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 *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 449–475, 2013.

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short Group Signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 41–55, 2004.

BC16. Olivier Blazy and Céline Chevalier. Structure-preserving smooth projective hashing. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*, pages 339–369, 2016.

BF01. Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference*, pages 213–229, 2001.

BF14. Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 520–537, 2014.

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

BH15. Mihir Bellare and Viet Tung Hoang. Adaptive Witness Encryption and Asymmetric Password-Based Cryptography. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 308–331, 2015.

Can01. Ran Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 136–145, 2001.

CFPZ09. Céline Chevalier, Pierre-Alain Fouque, David Pointcheval, and Sébastien Zimmer. Optimal Randomness Extraction from a Diffie-Hellman Element. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, pages 572–589, 2009.

COR99. Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional Oblivious Transfer and Timed-Release Encryption.

In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 74–89, 1999.

CS98. Ronald Cramer and Victor Shoup. A Practical Public Key Cryptosystem Provably Secure Against Adaptive Chosen Ciphertext Attack. In *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, pages 13–25, 1998.

CS02. Ronald Cramer and Victor Shoup. Universal Hash Proofs and a Paradigm for Adaptive Chosen Ciphertext Secure Public-Key Encryption. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, pages 45–64, 2002.

CZ14. Yu Chen and Zongyang Zhang. Publicly evaluable pseudorandom functions and their applications. In *Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings*, pages 115–134, 2014.

EHK$^+$13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge L. Villar. An algebraic framework for diffie-hellman assumptions. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 129–147, 2013.

FNV17. Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I*, pages 121–150, 2017.

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 *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 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 *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 518–535, 2014.

GGSW13. Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness Encryption and its Applications. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476, 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 *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 536–553, 2013.

GL06. Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange[1]. *ACM Trans. Inf. Syst. Secur.*, 9(2):181–234, 2006.

GLW14. Craig Gentry, Allison B. Lewko, and Brent Waters. Witness Encryption from Instance Independent Assumptions. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 426–443, 2014.

GM84. Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 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 *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–432, 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):1364–1396, 1999.

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

Jar14. Stanislaw Jarecki. Practical Covert Authentication. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 611–629, 2014.

JL09. Stanislaw Jarecki and Xiaomin Liu. Private Mutual Authentication and Conditional Oblivious Transfer. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 90–107, 2009.

KD04. Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 426–442, 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 *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 590–609, 2009.

KTY07. Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Group Encryption. In *Advances in Cryptology - ASIACRYPT 2007, 13th International Conference on the Theory and Application of Cryptology and Information Security*, pages 181–199, 2007.

KV11. Jonathan Katz and Vinod Vaikuntanathan. Round-Optimal Password-Based Authenticated Key Exchange. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 293–310, 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.

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 *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473, 2005.

Wee10. Hoeteck Wee. Efficient Chosen-Ciphertext Security via Extractable Hash Proofs. In *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, pages 314–332, 2010.

Zha16. Mark Zhandry. How to Avoid Obfuscation Using Witness PRFs. In *Theory of Cryptography - 13th International Conference, TCC 2016-A*, volume 9563 of *LNCS*, pages 421–448. Springer, 2016.