Practical Witness Encryption for Algebraic Languages And How to Reply an Unknown Whistleblower

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 seeming paradox. 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. But how should one *confidentially* reply to the *unknown* (anonymous) whistleblower. In brief, 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_R with witness relation R so that $L_R = \{x \mid \exists\, w : R(x, w) = 1\}$. The

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

encryption algorithm takes an alleged word x from the language L_R (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 the message m. Decryption only works if $x \in L_R$ and a ciphertext c computationally hides the message m if c has been computed with respect to some $x \notin L_R$.

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), i.e., graded encoding schemes. Later, Gentry et al. [GLW14] introduced the concept of positional WE, which allows to prove security of the aforementioned construction. 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 $x \notin L_R$, 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 $x \in L_R$. Thereby, the adversary as well as the extractor additionally get an auxiliary input z. 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, to avoid obfuscation. That is, a witness PRF can be used to construct WE in a straightforward way. Moreover, Zhandry also proposes (CCA secure) *reusable* WE, which introduces an additional global setup and thus allows to reuse certain parameters and consequently drastically reduces the size of ciphertexts in WE schemes. We observe that our generic constructions of WE bears 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. As all these constructions build upon MLMs and/or obfuscation, unfortunately, they are far from being practical. To this end, Abusalah et al. [AFP16] very recently introduced the notion of *offline* WE as a step towards making WE more practical. 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 an MLM/obfuscation candidate which is secure under some reasonable assumption.

Restricting Languages. In concurrent and independent work to ours, 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. PAoKs require a particular notion of knowledge extraction. 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

[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.

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 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. Moreover, 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 discussed 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 the use of 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, already in the seminal work [GGSW13] it has been shown 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 for inefficiency. We aim to make WE practical, but in contrast to offline WE 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 non-interactive Groth-Sahai (GS) proof framework [GS08]. Moreover, these languages are compatible with smooth projective hash functions (SPHFs). As we will see soon, a combination of SPHFs and the GS proof framework yield an interesting novel tool, which we apply to exemplarily solve the seemingly paradox discussed below.

A Motivating Question and Application. An interesting question that is motivated by the revelations of Edward Snowden is whether it is possible for *anyone to confidentially reply to an anonymous whistleblower*.[2] Let us assume that our whistleblower 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. However, the whistleblower has a strong interest in staying anonymous, i.e., identifying him could lead to severe consequences. To solve this dilemma, Rivest et al. [RST01] fortunately 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.[3] The scenario described above is illustrated in Figure 1.

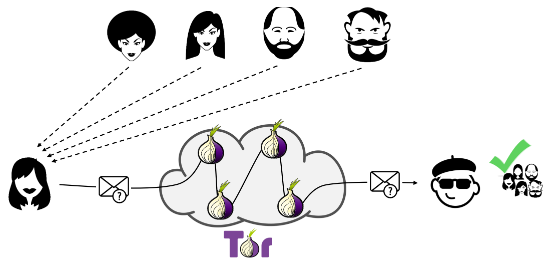

Fig. 1. Whistleblower Edwarda signs a document to be leaked using a ring signature and communicates it to a journalist. One possibility to realize this communication is via an anonymous bulletin board implemented by means of a Tor [tor] hidden service. After having received the message, the journalist knows that one of the five potential singers have leaked the document.

Given such a ring signature with a potentially huge ring size, 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

[2] Glenn Greenwald in his book [Gre14], for instance, describes how complicated it has been back in 2013 to get in touch and communicate with the (back then) anonymous source who claimed to have astonishing evidence of pervasive government spying and insisted to communicate via encrypted channels.

[3] Interestingly, [HO05] discuss that a paper-based analogue of ring signatures has already been used in Japan in the 18th century to demonstrate solidarity without leaking the initiator.

guaranteeing that only the whistleblower who has produced the ring signature can decrypt the reply. Such a scenario is depicted in Figure 2.

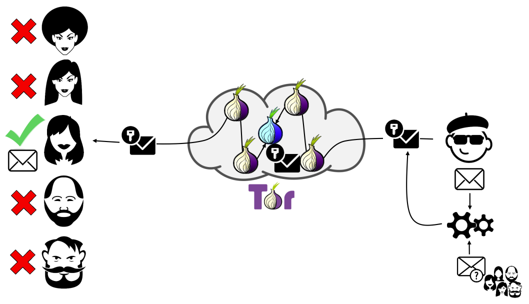

Fig. 2. The journalist publishes his reply (encrypted with respect to the ring signature) at the anonymous bulletin board. Then, anyone can potentially retrieve it, but only Edwarda is able to decrypt and thus read the reply. Thereby, the journalist does not know who of the five potential receivers will be able to read the reply.

As we will see later, a solution to the above dilemma can elegantly be achieved using the techniques proposed in this paper. In particular, applying our approach to WE to the GS proof technique and encrypting messages with respect to GS proofs. Basically, the idea is simply to use WE to encrypt a message with respect to a proof statement being part of the ring signature. The nice thing thereby is, that the so obtained solution is entirely practical.

1.2 Our Contribution

– We provide a generic construction of WE from SPHFs and prove (cf. Theorem 1) that if there exists an SPHF for a language L_R, then there exists an adaptively sound WE scheme for language L_R. Thereby, we define WE so that it provides an additional setup algorithm as it was already done in [AFP16, Zha16], since this notion makes the schemes more efficient and it is also more convenient to use in protocol design. While the relation between SPHFs and WE (without setup) was already informally mentioned in the literature, we believe that it is important to initiate a formal study.[4]
– Using standard techniques such as universal hashing and secure symmetric encryption schemes, we obtain a WE scheme for messages of arbitrary length.
– We present practical concrete instantiations of our generic approach to WE for algebraic languages in the bilinear group setting under the DLIN assumption. We, thereby, achieve compatibility with Groth-Sahai commitments (and thus statements from this proof system). Our approach is also easily portable

[4] Since we achieve statistically sound WE, it is known that using our approach it is impossible to construct WE for an **NP**-complete language (unless the polynomial hierarchy does collapse) as shown in [GGSW13].

to the SXDH setting (and thus relying on DDH). Besides being practically efficient, our constructions also only require standard assumptions.
- We illustrate that our generic approach to WE can be elegantly used to encrypt messages with respect to NIZK/NIWI proofs for statements in the GS proof system. Since the GS proof system is a frequently used building block in various cryptographic protocols, this yields a novel way of encryption and we assume that there are many interesting applications that could benefit from our technique.
- To illustrate the aforementioned concept, we present a concrete solution to the above problem in context of whistleblowing. We do so by showing how *anyone* getting to hold a valid ring signature can use our techniques to send a confidential message to the anonymous signer (whistleblower). This can be done in a way that the sender *does not need to know the whistleblower* (and thus no encryption key), but *no one except the whistleblower can decrypt.* As ring signatures in a bilinear group setting can generically be built using conventional signature schemes and non-interactive (witness indistinguishable) proof systems (cf. [Gha13]), we thus obtain a generic solution to this problem. In addition to the whistleblowing scenario, we also sketch two other applications, i.e., ring encryption and mutually-anonymous key-exchange.

1.3 Related Work

SPHFs (denoted as hash proof systems) were initially (implicitly) 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 are going to use it). The elegant idea in [CS98] is to combine ElGamal encryption with an 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. We note that this paradigm can be viewed as an implicit construction of publicly evaluable pseudorandom functions [CZ14].

Hybrid Encryption. Kurosawa and Desmedt [KD04] then discovered that the paradigm described above is also useful 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_R 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 an SPHF key pair $(\mathsf{hk}_i, \mathsf{hp}_i)$ for an SPHF defined for a language L_R, where $(C_i, p) \in R$ if C_i is a commitment to p. Membership in L_R 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]. In a follow up work it was very recently [BC16] 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 what we want to achieve as we require the GS proofs to be public and they must not be useful reconstruct the hash value. Furthermore, we note that other follow-up work on various aspects exists.

Group Encryption. Group encryption, introduced by Kiayias et al. in [KTY07], is the encryption analogue to group signatures. In group signatures any member of a managed group can anonymously produce signatures on behalf of the group. In case of a dispute, however, some dedicated authority (the opening authority) can reveal the identity of a signer. In group encryption, a sender can prepare a ciphertext and convince a verifier that it can be decrypted by a member of some managed group. Likewise to group signatures, in group encryption schemes an opening authority can reveal the identity of the group member that is capable of decrypting, if necessary. Consequently, group encryption (like group signatures) involves a dedicated trusted group manager and provides conditional anonymity, i.e, the trusted opening authority can break the anonymity.

Private/Covert Mutual Authentication. Private mutual authentication (also known as secret handshakes) [JL09] allows two parties belonging to some managed groups to privately authenticate and thereby protect the privacy of all authentication protocol inputs in the protocol. Covert mutual authentication [Jar14] is even stronger and allows two parties to authenticate to each other, but for everyone without a group membership certificate it is intractable to distinguish an instance of the protocol from a random beacon. The constructions in [JL09, Jar14] allow to covertly exchange an encryption key, but they are not ad-hoc nor unconditionally private. Every participant needs to obtain a group membership certificate (where the secret is even generated by the group manager) as it is the case in group signatures and group encryption and a group manager can recover the identity of any party involved in any protocol instance.

2 Background

Below, we provide the necessary background and recall some required primitives.

Notation. Let $x \xleftarrow{R} X$ denote the operation that picks an element x uniformly at random from X and let $x \in_R X$ denote that a value x is uniformly random in 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 $k > k_0$. In the remainder of this paper, 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. We call $e\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ a bilinear map or pairing if it is efficiently computable and the following conditions hold:*

Bilinearity: $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a) \quad \forall\, a, b \in \mathbb{Z}_p$
Non-degeneracy: $e(g, g) \neq 1_{\mathbb{G}_T}$*, i.e.,* $e(g, g)$ *generates* $\mathbb{G}_T$*.*

We use boldface letters for elements in $\mathbb{G}_T$, e.g., $\mathbf{g} = e(g, g)$. The symmetric (Type-1) setting as presented above is in contrast to the asymmetric setting (Type-2 or Type-3), where 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 using symmetric pairings, it is, however, important to note that our results carry over to the (more efficient) 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 $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$ with $\mathsf{BG} = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ in the symmetric bilinear group setting. Thereby, p is a prime of bitlength κ and represents the common group order of the groups $\mathbb{G}$ and $\mathbb{G}_T$, e is a pairing and g is a generator of $\mathbb{G}$.*

Definition 3 (Decision Linear Assumption). *The DLIN assumption in $\mathbb{G}$ states that for all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ there is a negligible function $\epsilon(\cdot)$ such that:*

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

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*

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

For our security proofs, we require that the output of such a hash function is "sufficiently" random if the input is "sufficiently" random. The leftover hash lemma provides the required arguments.

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, however, do not explicitly model a key generation algorithm and treat the keys as uniformly random bitstrings of length $\ell_{\Sigma,\kappa}$. Here, $\ell_{\Sigma,\kappa}$ is the keylength for encryption scheme Σ and security parameter κ.

Definition 5 (Symmetric Encryption Scheme). *A symmetric encryption scheme Σ is a tuple $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ of PPT algorithms which are defined as follows:*

$\mathsf{Enc}(k, m)$: *The encryption algorithm on input of a key k and a message m outputs a ciphertext c.*

$\mathsf{Dec}(k, c)$: *The decryption algorithm on input of a key k and a ciphertext c outputs a message m or $\perp$.*

We require Σ to be correct and $\mathsf{IND\text{-}T}$ secure, where $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA2}\}$. The respective definitions are provided below.

Definition 6 (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* $\Pr[\mathsf{Dec}(k, \mathsf{Enc}(k, m)) = m] = 1$.

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

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

where $\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA2}\}$ and $|m|$ is used to denote the length of message m. The oracle $\mathcal{O}_\mathsf{T}$ is defined as

$$\mathcal{O}_\mathsf{T} := \begin{cases} \mathcal{O}^{\mathsf{Enc}(k,\cdot)} & \text{if } \mathsf{T} = \mathsf{CPA} \\ \mathcal{O}^{\mathsf{Enc}(k,\cdot)}, \mathcal{O}^{\mathsf{Dec}(k,\cdot)} & \text{if } \mathsf{T} = \mathsf{CCA2}, \end{cases}$$

where Q^{Dec} denotes the list of queries to $\mathcal{O}^{\mathsf{Dec}}$ and we set $Q^{\mathsf{Dec}} \leftarrow \emptyset$ if $\mathsf{T} = \mathsf{CPA}$.

2.1 Smooth Projective Hashing

Smooth projective hash functions (SPHFs) can be seen as families of hash functions $\{H_{\mathsf{hk}}\}_{\mathsf{hk}\in K}$ with domain X, some associated language $L_R \subset X$, range $\mathsf{R} \subseteq \{0,1\}^n$ and key space K. The secret hashing key hk allows to compute a hash value for every $x \in X$. There is a second method for computing the hash value using a public projection key hp (computed from hk and possible dependent on the word x), which besides x also requires a witness w for membership of x in L_R, i.e., w such that $R(x,w) = 1$, to compute the hash value. The initial definition of SPHFs [CS02] requires that the projection key hp does not depend on the word x. Later, an alternative notion [KV11] was introduced, where hp may be word dependent. To obtain the most general result, we use the notion of [KV11], since one can simply assume that ProjKG ignores the word x for schemes adhering to [CS02]. Subsequently, we provide a formal definition of SPHFs.

Definition 8 (Smooth Projective Hash Function). *A SPHF for a language L_R and corresponding* **NP***-relation R is defined by the following PPT algorithms:*

$\mathsf{Setup}(1^\kappa)$: *This algorithm takes a security parameter κ and outputs the system parameters pp (including the description of the language L_R).*
$\mathsf{HashKG}(\mathsf{pp})$: *This algorithm takes the system parameters including a language L_R, and outputs a hashing key hk for L_R.*
$\mathsf{ProjKG}(\mathsf{hk}, x)$: *This algorithm takes a hashing key hk and a word x, and outputs a projection key hp (possibly depending on x).*
$\mathsf{Hash}(\mathsf{hk}, x)$: *This algorithm takes a hashing key hk and a word x, and outputs a hash H.*
$\mathsf{ProjHash}(\mathsf{hp}, x, w)$: *This algorithm takes a projection key hp, a word x, and a witness w for $x \in L_R$, and outputs a hash H.*

We assume that pp as well as L_R is implicitly contained in hk and hp, respectively.

For security an SPHF is required to be correct, smooth and pseudo-random. Below, we formally define these properties.

Definition 9 (Correctness). *A SPHF for a language L_R is correct, if for all κ, for all $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$, for all $x \in L_R$, for all w such that $R(x,w) = 1$, for all $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp})$, and for all $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, x)$, it holds that*

$$\mathsf{Hash}(\mathsf{hk}, x) = \mathsf{ProjHash}(\mathsf{hp}, x, w).$$

Definition 10 (Smoothness). *A SPHF for a language L_R is smooth, if for any $x \notin L_R$ it holds that:*

$$\begin{aligned}\{(L_R, \mathsf{pp}, x, \mathsf{hp}, H) \mid \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa), \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp}),\\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, x), H \leftarrow \mathsf{Hash}(\mathsf{hk}, x)\} \approx\\ \{(L_R, \mathsf{pp}, x, \mathsf{hp}, H) \mid \mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa), \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pp}),\\ \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, x), H \overset{R}{\leftarrow} \mathsf{R})\}\end{aligned}$$

where $\approx$ denotes statistical indistinguishability.

Additionally, for SPHFs we require *pseudo-randomness* as introduced by Gennaro and Lindell [GL06]. It requires that for any $x \in L_R$ without knowing a corresponding witness w, the distributions considered in Definition 10 remain computationally indistinguishable. A *secure* SPHF satisfies all the above properties.

If we have that for the domain X and the language $L_R \subset X$ the *subset membership problem* is hard (this means that it is hard to distinguish between a random element in L_R and a random element $x \in X \setminus L_R$) it is easy to show using a standard hybrid argument that smoothness implies pseudo-randomness.

2.2 Groth-Sahai (GS) Non-Interactive Zero-Knowledge Proofs

Groth and Sahai [GS08, GS07] provide a framework for efficient non-interactive witness-indistinguishable (NIWI) and zero-knowledge (NIZK) proofs for the satisfiability of various types of equations defined over bilinear groups. While the framework is quite independent of the underlying hardness assumption, we will use the instantiation based on the DLIN assumption, and, thus, our further explanations are tailored to this setting. We will focus on proofs for the satisfiability of pairing product equations (PPEs), which are of the form as in Equation 1

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

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. From an abstract point of view, GS proofs use the following strategy. 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 $F : \mathbb{G}^3 \times \mathbb{G}^3 \rightarrow \mathbb{G}_T^9$.

More formally, 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. Henceforth, let BG denote the description of the used bilinear group and let R be the relation such that $(\mathsf{BG}, x, w) \in R$ iff w is a satisfying witness for x with respect to BG. Furthermore, let L_R be the corresponding language.

Formally, a non-interactive proof system in a bilinear group setting is defined as follows:

Definition 11 (Non-Interactive Proof System). *A non-interactive proof system Π is a tuple $\Pi = (\mathsf{BGGen}, \mathsf{CRSGen}, \mathsf{Proof}, \mathsf{Verify})$ of PPT algorithms which are defined as follows:*

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

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

$\mathsf{Proof}(\mathsf{BG}, \mathsf{crs}, x, w)$: *This algorithm 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.

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

The GS proof system is perfectly complete, perfectly sound, and witness indistinguishable. Depending on the proven statement, it is can also be composably zero-knowledge. Since we do neither explicitly require these security properties for our illustrations nor for our security proofs, we refer the reader to [GS08] for formal definitions.

2.3 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 12 (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].

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.[5]

Definition 13. *A* WE *scheme defined for an* **NP***-language* L_R *with corresponding witness-relation* R *is a tuple* $\mathsf{WE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *of PPT algorithms which are defined as follows:*

$\mathsf{Gen}(1^\kappa)$: *This algorithm takes a security parameter* κ *and outputs public parameters* pp *(including a description of a language* L_R*).*

$\mathsf{Enc}(\mathsf{pp}, x, m)$: *This algorithm takes public parameters* pp*, some word* x *and a message* m *as input and outputs a ciphertext* c*.*

$\mathsf{Dec}(w, c)$: *This algorithm takes a witness* w *and a ciphertext* c *as input and outputs a message* m *or* $\perp$*.*

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

Definition 14 (Correctness). *A* WE *scheme for a language* L_R *is correct, if there exists a negligible function* $\epsilon(\cdot)$ *such that for all* κ*, for all* $\mathsf{pp} \leftarrow \mathsf{Gen}(1^\kappa)$*, for all* m*, for all* $x \in L_R$*, and for all witnesses* w *such that* $R(x, w) = 1$*, it holds that*

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

If $\epsilon = 0$*, we have perfect correctness.*

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

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

Remark 1. We note that assuming soundness of the WE scheme and that the subset-membership problem is hard for domain X and language $L_R \subset X$, i.e., the probability of a distinguisher is bound by $\epsilon(\kappa)$, one can use a standard hybrid argument to show that the probability to break soundness for $x \in L_R$ is bounded by $\epsilon(\kappa)$.

Definition 16 (Adaptive Soundness). *A* WE *scheme for a language* L_R *is adaptively sound, if for all PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that*

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

[5] Furthermore, instantiating WE from SPHFs *without a setup*, as informally sketched in [ABP15], could turn out to be involved, as this would require to provide a word x to the encryption algorithm before the description of the language is fixed.

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 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 of the SPHF. Now, if $m = 0$, then the ciphertext is a randomly sampled element from the range R, 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 (which is also included in the ciphertext) and consequently to decide whether $m = 0$ or $m = 1$ has been encrypted.

We stress that the construction paradigm used in our schemes inherently requires that the size of the SPHF's range $|\mathsf{R}|$ grows superpolynomial in the security parameter κ, but want to emphasize that this is the case for all existing SPHFs. We note that the smoothness definition of the SPHF already requires

$\mathsf{Gen}(1^\kappa)$: 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, $m \in \{0, 1\}$, $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp})$ and $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$. If $m = 0$, set $C \xleftarrow{R} \mathsf{R}$ and set $C \leftarrow H$ for $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, x)$ otherwise. Finally, return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$.

$\mathsf{Dec}(w, c)$: On input of w and c, parse c as $(C, x, \mathsf{hp}, \mathsf{pp})$ and compute $H \leftarrow \mathsf{SPHF.ProjHash}(\mathsf{hp}, x, w)$. Return 1 if $H = C$ and 0 otherwise.

Scheme 1: WE scheme for bits from SPHFs

that one can efficiently sample uniformly random elements from R.

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 $m = 0$ 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 $2^{-|\mathsf{R}|}$. □

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, $m \in \{0, 1\}$, run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$, $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, x)$. Sample $\boxed{C \xleftarrow{R} \mathsf{R}}$ and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$.

Transition - Game 0 $\rightarrow$ 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. $\square$

Remark 2. When constructing WE from SPHFs, the pseudo-randomness of the SPHF directly states that soundness of the WE scheme also holds computationally for $x \in L_R$ as long as no witness is known.

3.2 Extension to Messages of Arbitrary Length

While one could easily extend the WE scheme above to any message space $\mathcal{M} = \{0,1\}^\ell$ for any $\ell > 1$ by simply calling Enc independently for every message bit, we are looking for more efficient and compact solutions. Thus, we follow a standard paradigm in hybrid encryption. In Scheme 2 we present a construction that besides an SPHF requires a universal hash function family $\mathcal{H}$ and an at least $\mathsf{IND\text{-}CPA}$ secure symmetric encryption scheme Σ. The construction is quite straightforward. 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 $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^\kappa)$ and return pp.

$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}^*$, run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$ and $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, x)$, where $p(\cdot)$ is a polynomial defined by the concrete instantiation. Then randomly choose a universal hash function $\mathsf{H} : \mathsf{R} \rightarrow \{0,1\}^{\ell_{\Sigma,\kappa}}$ from the family $\mathcal{H}$, 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 $(C, x, \mathsf{hp}, \mathsf{pp}, \mathsf{H})$, compute $k \leftarrow \mathsf{H}(\mathsf{SPHF.ProjHash}(\mathsf{hp}, x, w))$, compute and return $m \leftarrow \Sigma.\mathsf{Dec}(k, C)$.

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

Theorem 2. *If SPHF is correct and smooth, $\mathcal{H}$ is a family of universal hash functions $\mathsf{H} : \mathsf{R} \rightarrow \{0,1\}^{\ell_{\Sigma,\kappa}}$, the symmetric encryption scheme Σ is correct and at least $\mathsf{IND\text{-}CPA}$ secure, and $p(\cdot)$ is such that $2^{(\ell_{\Sigma,\kappa} - |\mathsf{R}|)/2}$ 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:

$\mathsf{Enc}(\mathsf{pp}, x, m)$: On input of pp, x, $m \in \{0,1\}^*$, run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$ and $\boxed{H \xleftarrow{R} \mathsf{R}}$. Then randomly choose a universal hash functions $\mathsf{H} : \mathsf{R} \rightarrow \{0,1\}^{\ell_{\Sigma,\kappa}}$ from an appropriate family $\mathcal{H}$, 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, $m \in \{0,1\}^*$, run $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(\mathsf{pp})$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$. Choose $\boxed{k \xleftarrow{R} \{0,1\}^{\ell_{\Sigma,\kappa}}}$, 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 $2^{(\ell_{\Sigma,\kappa} - |\mathsf{R}|)/2}$. 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 $\mathsf{IND\text{-}T}$ ($\mathsf{T} \in \{\mathsf{CPA}, \mathsf{CCA}\}$) 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 $\mathcal{C}$, obtains the challenge ciphertext from $\mathcal{C}$ and returns it to the adversary. 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 $\mathsf{IND\text{-}T}$ 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. Consequently, such constructions often trade maximum efficiency for UC security. We do not aim for UC compatibility, as we focus on constructing WE and thus we strive for particularly efficient instantiations. Besides efficiency, our goal is to allow a maximum expressiveness of the language underlying the SPHF to ensure maximum flexibility in the choice of the form of the "private keys", i.e., the witnesses, and to also facilitate a broad range of possible applications. With these two goals in mind, it seems to be a nice tradeoff to restrict ourselves to the set of languages defined over bilinear groups, i.e., languages expressible via a set of PPEs.

4.1 SPHF for Linear Encryptions

As a basis, we use the ElGamal-based SPHF by Gennaro and Lindell [GL06], which we port to the DLIN setting (similar as it is done for linear Cramer-Shoup in [BBC+13a]). 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 $\mathsf{IND\text{-}CPA}$ security under the DLIN assumption. It is well known that such a scheme represents a perfectly binding and computationally hiding commitment scheme.

In Scheme 3, we present the SPHF, where the language L_R is with respect to the linear encryption public key pk contained in pp and contains all triples $(C_M, M) \in \mathbb{G}^3 \times \mathbb{G}$ of valid ciphertexts C_M with respect to pk and corresponding messages M. 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 $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{pk})$.

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

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

$\mathsf{Hash}(\mathsf{hk}, x)$: On input of $\mathsf{hk} = (\mathsf{pp}, L_R, \eta, \theta, \zeta)$ and $x = (\mathsf{pk}, C_M, M) \in \mathbb{G}^2 \times \mathbb{G}^3 \times \mathbb{G}$, where $C_M = (u, v, e)$, compute and return $H \leftarrow u^\eta v^\theta ({}^{e}/_{M})^\zeta$.

$\mathsf{ProjHash}(\mathsf{hp}, x, w)$: 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 3: SPHF for the language of linear ciphertexts

Lemma 2. *If the DLIN assumption holds, then the* SPHF *in Scheme 3 is secure.*

Proof (Correctness). Let L_R be the language of linear encryptions and pp, hk and hp be generated according to the setup in Scheme 3. Now, let $C_M = (\mathsf{pk}_1^{r_1}, \mathsf{pk}_2^{r_2}, M \cdot g^{r_1+r_2})$ be a linear encryption of some message M and $w = (r_1, r_2)$. Let $H_{\mathsf{Proj}} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, x, w)$ and $H_{\mathsf{Hash}} \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$, 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_R. 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, M) is in fact in L_R. Conversely, if $(C_M, M) \notin L_R$, we have $r_3 \neq r_1 + r_2$ and the value H looks perfectly random. $\square$

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 compute the ciphertext to M as $(g_1^r, g_2^s, M \cdot g^t)$, 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 pseudo-randomness game, whereas we have a distribution as in the smoothness game if the DLIN instance is random. Assuming the hardness of DLIN contradicts the existence of such an efficient distinguisher. $\square$

4.2 Extending Supported Languages

Applying the techniques presented in [BBC+13a, BBC+13b] to the SPHF in Scheme 3 allows us to extend the set of supported languages to a large class of PPEs. In particular, we can construct SPHFs for languages for the satisfiability of PPEs of the form

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

where X_i, Y_i and $\mathbf{Z}_i$ remain secret and are encrypted using linear encryption. However, as the application in Section 5.2 which we use to illustrate our techniques does not require the expressiveness of Equation 2, we use the following simplified equation for the ease of presentation:

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

Note that in a Type-1 setting, this simplification does not even influence the expressiveness. We further denote the commitments to Y_i as $C_i = (u_i, v_i, e_i) = (\mathsf{pk}_1^{r_{i1}}, \mathsf{pk}_2^{r_{i2}}, Y_i \cdot g^{r_{i1}+r_{i2}}) \in \mathbb{G}^3$ for $1 \leq i \leq m$ and $\mathbf{C}_i = (\mathbf{u}_i, \mathbf{v}_i, \mathbf{e}_i) = (e(\mathsf{pk}_1, g)^{r_{i1}}, e(\mathsf{pk}_2, g)^{r_{i2}}, \mathbf{Z}_i \cdot e(g, g)^{r_{i1}+r_{i2}}) \in \mathbb{G}_T^3$ for $m < i \leq n$. The language L_R contains all tuples $(\mathsf{PPE}, (C_i)_{i \in [m]}, (\mathbf{C}_i)_{m < i \leq n})$ of pairing product equations PPE and commitments C_i and $\mathbf{C}_i$, respectively. Membership in L_R is witnessed by the randomness used in the commitments.

For our following explanations, 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.

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

Lemma 3. *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 $m = 1, n = 2$. Let (r_{11}, r_{12}) and (r_{21}, r_{22}) be the randomness used to compute the linear encryptions of Y_1 and $\mathbf{Z}_2$, respectively. Then, (r_{11}, r_{12}) and (r_{21}, r_{22}) represent the witness. The projective hash value obtained using hp is computed as

$$\mathbf{H}_{\mathsf{Proj}} \leftarrow \prod_{i=1}^{1} e(A_i, \mathsf{hp}_{i1}^{r_{i1}} \mathsf{hp}_{i2}^{r_{i2}}) \cdot \prod_{i=2}^{2} e(g^{\gamma_i}, \mathsf{hp}_{i1}^{r_{i1}} \mathsf{hp}_{i2}^{r_{i2}}) =$$
$$e(A_1, (\mathsf{pk}_1^{\eta_1} g^{\zeta})^{r_{11}} (\mathsf{pk}_2^{\theta_1} g^{\zeta})^{r_{12}}) \cdot e(g^{\gamma_2}, (\mathsf{pk}_1^{\eta_2} g^{\zeta})^{r_{21}} (\mathsf{pk}_2^{\theta_2} g^{\zeta})^{r_{22}}).$$

Computing the hash value using hk yields:

$$\mathbf{H}_{\mathsf{Hash}} \leftarrow \mathbf{B}^{-\zeta} \cdot \prod_{i=1}^{1} e(A_i, u_i^{\eta_i} v_i^{\theta_i} e_i^{\zeta}) \cdot \prod_{i=2}^{2} (\mathbf{u}_i^{\eta_i} \mathbf{v}_i^{\theta_i} \mathbf{e}_i^{\zeta})^{\gamma_i} =$$
$$\mathbf{B}^{-\zeta} \cdot e(A_1, \mathsf{pk}_1^{r_{11}\eta_1} \mathsf{pk}_2^{r_{12}\theta_1} (Y_1 \cdot g^{r_{11}+r_{12}})^{\zeta}) \cdot e(g, \mathsf{pk}_1)^{r_{21}\eta_2\gamma_2} \cdot e(g, \mathsf{pk}_2)^{r_{22}\theta_2\gamma_2} \cdot$$
$$\mathbf{Z}_2^{\zeta\gamma_2} \cdot e(g, g)^{(r_{21}+r_{22})\zeta\gamma_2} =$$
$$\mathbf{B}^{-\zeta} \cdot e(A_1, Y_1^{\zeta}) \cdot \mathbf{Z}_2^{\zeta\gamma_2} \cdot e(A_1, \mathsf{pk}_1^{r_{11}\eta_1} \mathsf{pk}_2^{r_{12}\theta_1} (g^{r_{11}+r_{12}})^{\zeta}) \cdot e(g, \mathsf{pk}_1)^{r_{21}\eta_2\gamma_2} \cdot$$
$$e(g, \mathsf{pk}_2)^{r_{22}\theta_2\gamma_2} \cdot e(g, g)^{(r_{21}+r_{22})\zeta\gamma_2} \overset{(ii)}{=}$$
$$e(A_1, (\mathsf{pk}_1^{\eta_1} g^{\zeta})^{r_{11}} (\mathsf{pk}_2^{\theta_1} g^{\zeta})^{r_{12}}) \cdot e(g^{\gamma_2}, (\mathsf{pk}_1^{\eta_2} g^{\zeta})^{r_{21}} (\mathsf{pk}_2^{\theta_2} g^{\zeta})^{r_{22}}).$$

where for the last step (ii), 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 2) a distinguisher for DLIN. □

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

4.3 SPHF for Linear Groth-Sahai Commitments

Now, let the language for the SPHF be defined by the commitments used within the GS proof framework. This brings us closer to our final technique, i.e., to construct WE such that we can reuse commitments, which were initially used in a GS proof. Therefore, we subsequently show how to extend the SPHF in Scheme 3 to work with linear GS commitments [GS08]. Before we do so, we introduce some additional notation. Let $\circ : \mathbb{G}^{1\times n} \times \mathbb{G}^{1\times n} \rightarrow \mathbb{G}^{1\times n}$ and $\cdot : \mathbb{Z}_p \times \mathbb{G}^{1\times n} \rightarrow \mathbb{G}^{1\times n}$ 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 $\log_g U_3 \notin \mathrm{span}(\log_g U_1, \log_g U_2)$. 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 4, we present the SPHF for linear GS commitments. Essentially, this scheme is a tweak of Scheme 3, where we need to extend the projection key hp due to g_1^{ρ}, g_2^{ν}, and $g^{\rho+\nu}$ being fixed in the public key of linear GS commitments (i.e., the CRS of the GS proof system).

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

Proof (Correctness). Let L_R be the language of linear GS commitments C_M and corresponding committed messages M and pp, hk and hp be generated according to the setup in Scheme 4. 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 $H_{\mathsf{Proj}} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, x, w)$ and $H_{\mathsf{Hash}} \leftarrow \mathsf{Hash}(\mathsf{hk}, x)$, then we have

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

□

$\mathsf{Setup}(1^\kappa)$: 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 $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{pk})$.
$\mathsf{HashKG}(\mathsf{pp})$: On input of pp, return $\mathsf{hk} \leftarrow (\mathsf{pp}, \eta, \theta, \zeta) \xleftarrow{R} \mathbb{Z}_p^3$.
$\mathsf{ProjKG}(\mathsf{hk}, x)$: On input of hk and some word $x = (C_M, M) \in \mathbb{G}^3 \times \mathbb{G}$, 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 $\mathsf{hp} \leftarrow (\mathsf{pp}, \mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_3) = (g_1^\eta g^\zeta, g_2^\theta g^\zeta, (g_1^\rho)^\eta (g_2^\nu)^\theta (g^{\rho+\nu})^\zeta)$.
$\mathsf{Hash}(\mathsf{hk}, x)$: On input of $\mathsf{hk} = (\mathsf{pp}, \eta, \theta, \zeta)$ and $x = (C_M, M) \in \mathbb{G}^3 \times \mathbb{G}$, where $C_M = (u, v, e)$, compute and return $H \leftarrow u^\eta v^\theta ({}^{e}/_{M})^\zeta$.
$\mathsf{ProjHash}(\mathsf{hp}, x, w)$: On input of hp, 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 4: SPHF for the language of linear GS commitments

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_R. 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 $\log_g H \in \operatorname{span}(\log_g \mathsf{hp}_1, \log_g \mathsf{hp}_2, \log_g \mathsf{hp}_3)$ is when $r_4 = (r_1 + \rho r_3) + (r_2 + \nu r_3) = r_1' + r_2'$, i.e., when (C_M, M) is in fact in L_R. Conversely, if $(C_M, M) \notin L_R$ we have that $r_3 \neq r_1' + r_2'$ and the value H looks perfectly random. ☐

One could now straight-forwardly show that pseudo-randomness holds under the hard subset membership problem in the soundness setting. However, switching to the unconditional hiding setting would render our argumentation meaningless, as for every possible message there exists a randomness so that a given commitment opens to this message in this setting. In other words, all words which are not in the language in the binding setting are in the language in the hiding setting. But based on this observation we can proof pseudo-randomness.

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 smoothness definition for some word $(C_M, M') \notin L_R$.
Distribution 1: As D^0, but we set up pk to be unconditionally hiding, i.e., $\mathsf{pk} = (g_1, g_2, g, g_1^\rho, g_2^\nu, \boxed{g^\psi})$ with $\psi \xleftarrow{R} \mathbb{Z}_p$.
Transition $\mathsf{D}^0 \rightarrow \mathsf{D}^1$: A distinguisher $\mathcal{D}^{0\rightarrow 1}$ is a DLIN distinguisher (more generally contradicts the CRS indistinguishability of GS).

In D^0 the hash value is statistically close to random. In D^1 there exists an opening for C_M to M', i.e., $(C_M, M') \in L_R$. Both distributions are computationally indistinguishable, which proves pseudo-randomness. $\square$

4.4 Extending Supported Languages II

We can now use the SPHF for linear GS commitments in statements over bilinear groups in a similar way as described for the plain DLIN ciphertexts in Section 4.2. In particular, 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

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

Security, of the construction in Equation (4) is easy to verify by the security of Scheme 4 using the strategy in Section 4.2 with the argumentation in Section 4.3. Thus, we omit the proof and directly state the lemma.

Lemma 5. *Using the* SPHF *in Scheme 4 as described above yields a secure* SPHF *for any language covered by Equation* (3).

5 Witness Encryption and GS Proofs

Below we show that plugging the SPHF in Scheme 4, used as demonstrated in Equation (4), into our generic WE constructions yields a methodology to encrypt a message with respect to a large class of statements over PPEs proven using GS (cf. Section 5.1). In Section 5.2 we show how to generically apply this technique to solve the problem from our motivation, i.e., to allow for a confidential reply to an unknown (anonymous) ring signer. Then, in Section 5.3 we provide a brief performance analysis of our technique to emphasize its practicality.

5.1 Encrypting With Respect to a GS Proof

To give a brief overview of our idea, assume that a prover creates a proof π for the satisfiability of some PPE. To conduct a GS proof, informally, one sends commitments to the witness together with some additional group elements that are used to "cancel out" the randomness in the commitments. Now, given such a proof, one can simply encrypt a message with respect to the statement proven in π using our SPHF in Scheme 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 5 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 4.3. We assume that GS.BGGen and 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. For simplicity we use PPEs of the form in Equation (3) without the $\mathbf{Z}_i$ values. However, recall that our techniques can straight forwardly be adapted to work with all statements covered by Equation (2). We write a GS proof π as a sequence of commitments

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 $((C_i)_{i\in[n]}, \mathsf{PPE}, \pi_{\mathsf{GS}})$,
		set $\mathsf{pp} \leftarrow (\mathsf{BG}, \mathsf{crs})$,
extract r_{com} from r	$\xleftarrow{c}$	$c \leftarrow \mathsf{WE.Enc}(\mathsf{pp}, ((C_i)_{i\in[n]}, \mathsf{PPE}), m)$
$m \leftarrow \mathsf{WE.Dec}(r_{\mathsf{com}}, c)$		

Scheme 5: Encryption of a message with respect to a GS proof

$(C_i)_{i\in[n]}$, 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, the language used in the WE scheme consists of words containing the PPE as well as the commitments $(C_i)_{i\in[n]}$ to the unrevealed values $(Y_i)_{i\in[n]}$. Membership in this language is witnessed by the randomness r_{com}.

To formally capture the security we would expect when using WE in this context, we require that breaking soundness remains intractable even if the statement is in fact in L_R and a GS proof for this fact is provided.

Definition 17 (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 defined by* $(\mathsf{BG}, \mathsf{PPE}, \mathsf{crs})$ *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*

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

Theorem 3. *When instantiating Scheme 5 with a* WE *scheme based on the* SPHF *from Equation* (4)*, and there is more than one possible witness for the statement, then Scheme 5 provides pseudo-randomness in the presence of proofs.*

Proof. Due to the CRS indistinguishability of GS we can without loss of generality assume the unconditionally hiding setting. By the pseudo-randomness of the SPHF in Equation (4) we know that soundness of the WE scheme also holds if $x \in L_R$ as long as no witness is known and no additional information is provided (recall that we do not require the hard-subset membership problem to prove pseudo-randomness). What, thus, remains is to show that the proof does not contain additional information: As soon as there exists more than one possible witness, the GS proof unconditionally hides the witness among all possible witnesses (cf. [GS07, Proof of Theorem 3]), which, in further consequence means that it unconditionally hides the used randomness r_{com} among all possible values. This proves the theorem. $\square$

Alternatively, one could also directly prove the above theorem in the uber-assumption framework [Boy08] using the soundness setting together with pseudo-randomness 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 $i > 1$.

5.2 How to Reply an Unknown Whistleblower

Now we come back to our motivating question. Recall, that we assumed that a whistleblower Edwarda who signs a leaked document using a ring signature, i.e., a signature which hides her identity unconditionally among other carefully selected people in an ad-hoc group without getting their approval or assistance. We asked ourselves if a journalist can confidentially reply the unknown whistleblower.

Indeed, we can use our techniques 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). Therefore, we first recall a generic construction of ring signatures for bilinear groups [Gha13].

A Generic Construction of Ring Signatures. Ghadafi [Gha13] 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) 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, which is reasonable in a discrete log setting. 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 6, 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, $F : \{0,1\}^* \to \mathcal{M}$ denotes a collision resistant hash function.

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

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]).

$\mathsf{Setup}(1^\kappa)$: 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 6: Generic construction of ring signatures [Gha13]

Theorem 4 ([Gha13]). *Scheme 6 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.*

Encrypting a Confidential Reply to an Unknown Receiver. For every ring signature scheme following the paradigm in Scheme 6, where π_{L_1} proves the satisfiability of a PPE that is compatible with our SPHF framework from Section 4.3, one can use the technique presented in Scheme 5 to encrypt messages with respect to π_{L_1}. The fact that the anonymity of ring signatures crucially requires that more than one witness exists for R_1 and from Theorem 3 we straight-forwardly derive the following corollary.

Corollary 1. *When using Scheme 5 to encrypt with respect to the π_{L_1}-parts of a ring signature, pseudorandomenss in the presence of proofs holds.*

In Appendix B we provide a concrete instantiation of Scheme 6 using Waters signatures [Wat05] as an example of a compatible ring signature scheme.

5.3 Efficiency of Our Technique

We want to emphasize that our technique is 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 4 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 finally underline the practicality of our construction with respect to our exemplary application from Section 5.2, we analyze the computational effort that is necessary to encrypt a message with respect to a ring signature based on Waters signatures (cf. Scheme 7 in Appendix B). 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.

6 Discussion

Finally, we briefly discuss two other potential applications of the presented methodologies and leave a rigorous investigation as future work.

Ring Encryption. It is quite straightforward to construct a ring encryption equivalent to group encryption [KTY07], i.e., an *ad-hoc* version of 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. A simple instantiation (in analogy to the generic ring signature construction) would assemble a ring $\mathcal{R}$ of public keys of a key-private public key encryption scheme [BBDP01], e.g., ElGamal or linear encryption, with shared group parameters. Then, one would take the receivers public key pk_i and encrypt the message m. Moreover, using a compatible non-interactive proof system (such as GS), one proves that pk_i is a member of the ring $\mathcal{R}$ without revealing pk_i (e.g., by using a membership proof similar as in [Gha13]). Our use of WE in Section 5.2 can be interpreted as a construction of a variant of *ring encryption* in the sense that we additionally achieve *unconditional anonymity.* 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 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 5.2.

Acknowledgements. We thank the anonymous referees from Crypto'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.

[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.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-Privacy in Public-Key Encryption. In *ASIACRYPT*, 2001.

[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. *IACR Cryptology ePrint Archive*, page 258, 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.

[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.

[Gre14] Glenn Greenwald. *No Place to Hide: Edward Snowden, the NSA, and the U.S. Surveillance State*. Metropolitan Books/Henry Holt (NY), 2014.

[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.

[HO05] Yoshikazu Hanatani and Kazuo Ohta. Two Stories of Ring Signatures. CRYPTO 2005 Rump Session Talk, https://www.iacr.org/conferences/crypto2005/r/38.ppt, 2005.

[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.

[tor] Tor project: Anonymity online.

[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 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* $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ *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 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 7, 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 $F : \{0,1\}^* \rightarrow \{0,1\}^k$. 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 7: Waters signature based ring signature scheme [Gha13]

Theorem 5 **([Gha13]).** *The construction in Scheme 7 is secure if the DLIN assumption holds in $\mathbb{G}$, F is a collision resistant hash function, and* GS *is sound and witness indistinguishable.*