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. It greatly extends the scope of encryption as it allows to encrypt a message using the description of a hard problem (a word in some language) and someone who knows a solution to this problem (a witness) is able to decrypt. Recent work thereby focuses on constructing WE for **NP**-complete languages (and thus obtaining WE for any language in **NP**). While this is an interesting challenge, it is also the main source for inefficiency and requires non-standard assumptions related to multilinear maps and obfuscation. We ask whether it is possible to come up with practically efficient WE schemes, which are still expressive enough to provide a solution to the following problem. Assume that an anonymous whistleblower, say Edwarda, wants to leak authoritative secrets in a way that the public will be convinced of its authenticity, but she wants to stay anonymous. Therefore, she signs the leaked document using a ring signature. Such a signature hides her identity unconditionally among other carefully selected people in an ad-hoc group and does not require getting their approval or assistance. But now the question arises as how to *confidentially* reply to such an *unknown* (anonymous) whistleblower.

In this paper we answer this question and introduce *practical* constructions of WE that are expressive enough to elegantly solve the seeming paradox sketched above. To this end, we restrict the class of supported languages from any **NP**-language to algebraic languages (defined over bilinear groups). In doing so, we obtain simple generic constructions, which only rely on smooth projective hash functions and can be instantiated from standard assumptions. Based on our generic constructions, we then show how to encrypt a message with respect to a given ring signature. Thereby, we only use information from a given ring signature (specifying an **NP**-language) such that only the anonymous signer behind the ring signature can decrypt (as *only* she holds the respective witness). In particular, we provide efficient instantiations for any ring signature scheme obtained from EUF-CMA-secure signature schemes and witness-indistinguishable Groth-Sahai proofs.

Keywords: Witness encryption, smooth projective hash functions, ring signatures, ring encryption, leaking secrets.

1 Introduction

Witness encryption (WE) is a recent powerful encryption paradigm introduced by Garg et al. [GGSW13a]. 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 encryption algorithm takes an alleged instance x of 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.

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

Decryption only works if x is actually in the language and a ciphertext c computationally hides a 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 [GGSW13a] has been for the **NP**-complete problem *exact cover*. It uses approximate multilinear maps (MLMs), but Garg et al. did not provide a reduction of the security of their instantiation to anything simpler than directly assuming its security. 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 implies the existence of an extractor that extracts the witness w for x used in the encryption. 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 [Zha14] 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 for concrete applications.

As all the above constructions build upon MLMs and/or obfuscation, they are far from being practical. To this end, Abusalah et al. [AFP15] very recently introduced the notion of *offline* WE as a step towards making WE more practical. Their approach is to split encryption into an expensive offline phase and a much more efficient online phase. Thus, they achieve practical efficiency for the online part (essentially a twin ElGamal encryption). Nevertheless, the offline part and the decryption still requires obfuscation and thus cannot be considered as being nearly practical. Besides imposing a relatively large 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. Thus, building schemes upon those primitives introduces a security risk which is hard to quantify.

In concurrent and independent work to ours, Faonio et al. [FNV15] introduced the concept of predictable arguments of knowledge (PAoK) systems. They are one-round interactive protocols in which the verifier generates a challenge (to be sent to the prover) and can at the same time predict the prover's answer to that challenge. Moreover, they 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 construct PAoKs from extractable hash proof systems (Ext-HPS) as defined 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 extractable hash proof systems 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

[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.

due to the lack in expressiveness of Ext-HPS as used in [FNV15], their constructions are not suitable for our application.

Finally, we note that earlier work on (private) conditional oblivious transfer [COR99, JL09] can be viewed as as an interactive version of (extractable) WE. Constructions are however, only presented for very specific and restricted languages not suitable for our applications.

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]. 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 [GGSW13a] it has been shown that WE can be used to construct identity-based encryption [BF01] as well as attribute-based encryption [SW05] for circuits.

1.1 Our Approach

While having WE schemes that support *all languages* in **NP** is appealing, it is the main source for inefficiency. In this paper we aim at making WE practical, but in contrast to offline WE we focus on all aspects, i.e., encryption as well as decryption. We thereby improve efficiency by restricting the class of languages supported by a WE scheme from any **NP**-language to languages compatible with smooth projective hash functions (SPHFs) and in particular to *algebraic languages defined over bilinear groups*. Such languages are very relevant for many practical applications 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 proof framework [GS08]. Since we, thereby, 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 [GGSW13a].

Our approach is quite straightforward and works as follows: We use a smooth projective hash function (SPHF) associated to a language L_R. An SPHF defines two hashing modes: One mode uses a secret hashing key hk to produce a hash for a value x and the other mode works with a public (projection) key, but besides x also requires a witness w to the membership of x in L_R. Now, loosely speaking, to encrypt a message $m \in \{0, 1\}$ with respect to some x (which may be in L_R or not), we run the SPHF setup to generate a secret hashing key and a public projection key. Then, if $m = 1$, we produce a hash value that represents the ciphertext. Otherwise, i.e., if $m = 0$, we sample a uniformly random element from the range of the SPHF as the ciphertext. Then, we additionally include the projection key in the ciphertext. Clearly, if $x \in L_R$ someone who knows a corresponding witness w can recompute the hash value using the projection key and thus determine whether $m = 0$ or $m = 1$. Thereby, the properties of the SPHF ensure that—independent of the bit m—the resulting hash value looks uniformly random in the range of the SPHF as long as no witness is known.

We provide a generic construction and prove (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. Moreover, using standard techniques such as universal hashing and secure symmetric encryption schemes, we obtain a WE scheme for messages of arbitrary length from any such WE scheme, i.e., by extracting an encryption key from the hash value. We also present practical concrete instantiations of our generic approach for algebraic languages in the bilinear group setting under the DLIN assumption, which is compatible with Groth-Sahai commitments (and thus statements from this proof system). Our approach is also easily portable to the SXDH setting. Besides being practically efficient, our constructions also only require standard assumptions.

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*. 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 on communicating only through encrypted channels.

Let us assume that our whistleblower wants to leak a secret to a journalist. Here we have conflicting interests. The journalist who receives the exposed information has a strong interest in the correctness of the information before publishing it, as this could destroy his career. One way to ensure this is to have a high level of confidence that the information indeed comes from an insider. However, a whistleblower clearly has a strong interest in staying anonymous, as identifying him could lead to severe consequences. Rivest et al. [RST01] came up with an elegant cryptographic primitive denoted as ring signature to solve this dilemma. In such a signature scheme, a signature hides the identity of the signer unconditionally among other people in an ad-hoc group 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.[2] Given such a ring signature with a potentially very large ring size an immediate question is how to privately reply to the unknown sender.

The whistleblower could append a public key for encryption to the document before producing the ring signature. However, this is dangerous for the journalist, as the whistleblower could publish the corresponding private decryption key without loosing her anonymity. This, in turn, might cause troubles for the journalist. Another naive idea would be to "misuse" the public verification keys of all ring members to encrypt the message under each public key in the ring. Obviously, this is a bad idea, as apart from yielding very large ciphertexts, this also means that not only the originator but all members of the ring could read the confidential reply. The arguably most elegant solution would be to directly use the ring signature to encrypt the reply to the whistleblower, while guaranteeing that only the whistleblower can decrypt the reply. Furthermore, it would be nice that publishing the "decryption key" will destroy the anonymity of the whistleblower (thus increasing the motivation not to publish the "decryption key").

It turns out that such a fair solution can elegantly be achieved using the approach to WE that we present in this paper. In particular, ring signatures in a bilinear group setting can generically be built using conventional signature schemes as well as non-interactive (witness indistinguishable) proof systems as demonstrated by Ghadafi [Gha13]. In Section 5, we show how *anyone* getting to hold such a valid ring signature can send a confidential message to the anonymous signer (whistleblower) 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*

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

using our approach to WE. Basically, the idea is simply to use WE to encrypt a message with respect to a proof (taken from a signature) for the proof statement in the ring signature scheme. The nice thing thereby is, that the approach is entirely practical.

We emphasize that recently research in methods to anonymously leak information has seen a renaissance and the study of new interesting concepts such as cryptogenography [BJSW14] and anonymous steganography [JO16] has been initiated. However, these solutions do not fit into our problem description because they either require heavy interaction or a small anonymous channel in the first place, respectively.

Additional Applications of our Approach. What makes our approach of constructing WE attractive, is that it allows us to generally encrypt messages with respect to statements (that have been proven) under the Groth-Sahai proof framework and thus we assume that there are many other interesting applications for our approach. In Section 6 we briefly discuss two other applications which may be of independent interest.

1.2 Related Work

SPHFs (denoted as hash proof systems) were initially (implicitly) used to construct IND-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] (essentially, 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. Here the public key includes a projection key of the SPHF and the secret key includes the corresponding hashing key. Roughly, encryption, besides producing an 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.

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 IND-CCA2 secure hybrid encryption schemes. Similar as in [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 [GL03] 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]. We note that further follow-up work on various aspects in this directions 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 (a.k.a. secret handshakes) [JL09] allows two parties being member of 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 any party not having a group membership certificate cannot even distinguish an instance of such 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

In this section, we provide the necessary background and recall some of the 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. 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. We use $[n]$ to denote the set $\{1, \ldots, n\}$.

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 lower-case 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., $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. 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, where such translations can already be nicely automated [AGH15].

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

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

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*

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

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 $\delta \geq -\log(\max_{x\in\mathcal{X}} \mathsf{Pr}[X = x])$ 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*

$$\frac{1}{2} \sum_{z\in\{0,1\}^\ell} \left|\mathsf{Pr}[\mathsf{H}_y(X) = z] - 2^{-\ell}\right| \leq 2^{(\ell-\delta)/2}.$$

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 is a tuple $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ of PPT algorithms which are defined as follows:*

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

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

Definition 6 (Correctness). *A symmetric encryption scheme 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*

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

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

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

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*

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

and $Q^{\mathsf{Dec}} = \emptyset$ *if* $\mathsf{T} = \mathsf{CPA}$*, which denotes the list of queries to* $\mathcal{O}^{\mathsf{Dec}}$ *if* $\mathsf{T} = \mathsf{CCA2}$.

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 a key space K. Having a secret hashing key hk, a hash value can be computed for any $x \in X$. 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, a stronger notion [KV11] was introduced, where hp may be word dependent. To obtain the most general result, we subsequently 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 (algorithmic) 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.

$\mathsf{HashKG}(L_R)$: *This algorithm takes 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.

$\mathsf{ProjHash}(\mathsf{hp}, x, w)$: *Takes a projection key* hp*, a word* x*, and a witness* w *for* $x \in L_R$ *and outputs a hash* H.

In the algorithms above, pp *generated by the* Setup *algorithm is implicitly available to all other algorithms, and* 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* $x \in L_R$*, for all* w *such that* $R(x, w) = 1$*, for all* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, for all* $\mathsf{hk} \leftarrow \mathsf{HashKG}(L_R)$*, 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). *Smoothness requires that for any $x \notin L_R$, it holds that:*

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

where $\approx$ denotes statistical indistinguishability.

As usual, we consider 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 $x \in X \setminus L_R$. This clearly implies the hardness of finding a witness w for a given word $x \in L_R$. In such a setting SPHFs need to provide an additional property denoted as *pseudo-randomness*. It requires that for any $x \in L_R$ without knowing a witness w for this membership, the distributions considered in Definition 10 remain computationally indistinguishable. We call a SPHF *secure* if it satisfies all the above properties.

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

Groth and Sahai [GS08] 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 being based on the DLIN setting, and, thus, our further explanations are tailored to this setting. Furthermore, we will focus on proofs for the satisfiability of pairing product equations (PPEs), which are of the form

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

where $(X_1, \ldots X_m) \in \mathbb{G}^m$, $(\hat{Y}_1, \ldots, \hat{Y}_n) \in \mathbb{G}^n$ are the secret vectors (to prove knowledge of) and $(A_1, \ldots, A_n) \in \mathbb{G}^n, (\hat{B}_1, \ldots, \hat{B}_m) \in \mathbb{G}^m, (\gamma_{ij})_{i\in[m],j\in[n]} \in \mathbb{Z}_p^{n\cdot m}$, 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. The proof π is used to "cancel out" the randomness used in the commitments. However, this does not directly work when using the groups $\mathbb{G}, \mathbb{G}$, and $\mathbb{G}_T$, but requires to project the involved elements to the vector spaces $\mathbb{G}^3, \mathbb{G}^3$, and $\mathbb{G}_T^9$ in the DLIN setting by using the defined projection maps and to prove the satisfiability of the PPEusing the projected elements and corresponding bilinear map $F : \mathbb{G}^3 \times \mathbb{G}^3 \to \mathbb{G}_T^9$.

More formally, a GS proof for a PPEallows 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 of statements in R.

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 of PPT algorithms* (BGGen, CRSGen, Proof, Verify)*, 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, and outputs* 1 *if* π *is valid and* 0 *otherwise.*

The GS proof system is perfectly complete, perfectly sound, and witness indistinguishable. Furthermore, it is composably zero-knowledge if $t_T = 1_{\mathbb{G}_T}$ and the PPE does not involve a pairing of two public constants. Since we do neither explicitly require any of 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 $\mathcal{R}$, everybody can verify the validity of such a signature with respect to $\mathcal{R}$, but it is infeasible to identify the actual signer. Subsequently, we provide a formal definition of ring signature schemes, which we adopt from [Gha13].

Definition 12. *A ring signature scheme 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* 1 *if* σ *is a valid signature on* m *with respect to* $\mathcal{R}$ *and* 0 *otherwise.*

For security, a ring signature scheme is required to be correct, unforgeable, and anonymous. Informally those requirements are defined as follows. Correctness requires that Verify accepts all honestly generated signatures. Unforgeability requires that only when being in possession of a secret key sk_i that corresponds to some public key $\mathsf{pk}_i \in \mathcal{R}$ one can issue valid signatures with respect to $\mathcal{R}$. Finally, 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

A witness encryption scheme [GGSW13a] is defined as below, where for security we use the stronger adaptive soundness notion introduced in [BH13, BH15]. For completeness, we also include the modified version [GGSW13b] of the soundness definition from [GGSW13a] (cf. [BH13]).

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

$\mathsf{Enc}(1^\kappa, x, m)$: *This algorithm takes a security parameter* κ*, some string* $x \in \{0,1\}^*$ *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 always assume that L_R *is an implicit input to both algorithms. While it is implicitly assumed in existing work, we explicitly require that the behavior of* Enc *must be independent of whether* $x \in L_R$ *or* $x \notin L_R$*.*

We require a witness encryption scheme 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* m*, for all* $x \in L_R$*, and for all witnesses* w *such that* $R(x, w) = 1$*, it holds that*

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

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

Definition 15 (Soundness). *A* WE *scheme is sound, if for any PPT adversaries* $\mathcal{A}$ *there is a negligible function* $\epsilon(\cdot)$ *such that for any* $x \notin L_R$ *and for any* m_0, m_1

$$|\Pr[\mathcal{A}(\mathsf{Enc}(1^\kappa, x, m_0)) = 1] - \Pr[\mathcal{A}(\mathsf{Enc}(1^\kappa, x, m_1)) = 1]| \leq \epsilon(\kappa)$$

where $|m_0| = |m_1|$*.*

Remark 1. We note that assuming soundness of the WE scheme and that the subset-membership problem is hard for domain X and language $L_R \subset X$, one can use a standard hybrid argument to show that the probability to break soundness for $x \in L_R$ is negligible.

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

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

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

3.1 Generic Construction of Bit WE from SPHFs

We are now ready to present our generic construction of a WE scheme from any SPHF. As a warm up, we start with a bit encryption WE scheme (cf. Scheme 1), i.e., we assume that the message space is $\mathcal{M} = \{0,1\}$. 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 range $|\mathsf{R}|$ of the SPHF grows with the security parameter κ provided to $\mathsf{SPHF.Setup}$, but want to emphasize that this is the case for all existing SPHFs.

$\mathsf{Enc}(1^\kappa, x, m)$: On input of κ, x, $m \in \{0, 1\}$, run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^\kappa)$, $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(L_R)$ 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 $H \leftarrow \mathsf{SPHF.ProjHash}(\mathsf{hp}, x, w)$. Return 1 if $H = C$ and 0 otherwise.

Scheme 1: WE scheme for bits from SPHFs

We note that the smoothness definition of the SPHF already requires 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). Let us analyze the probability that Scheme 1 is not correct, i.e., the probability that it happens 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:

$\mathsf{Enc}(1^\kappa, x, m)$: On input of κ, x, $m \in \{0, 1\}$, run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^\kappa)$, $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(L_R)$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$, $H \leftarrow \mathsf{SPHF.Hash}(\mathsf{hk}, x)$. Sample $\boxed{C \xleftarrow{R} \mathsf{R}}$ and return $c \leftarrow (C, x, \mathsf{hp}, \mathsf{pp})$.

Transition - Game 0 → Game 1: By the smoothness property of the SPHF, the adversary's view in Game 1 is statistically close to the view in Game 0, i.e., Game 0 and Game 1 are statistically indistinguishable.
Game 1 is simulated independent of the bit b and distinguishing it from Game 0 would imply a distinguisher for two statistically close distributions. □

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 encrypting the message bit by bit, i.e., calling Enc independently for every message bit, we are looking for more efficient and compact solutions. To this end, 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 and uses a universal hash function $H \in \mathcal{H}$ on the hash value of the SPHF as a randomness extractor to obtain an encryption key for Σ. We also note that for the languages we have in mind, i.e., group-dependent languages, one could also use alternative randomness extractors such as [CFPZ09].

$\mathsf{Enc}(1^\kappa, x, m):$ On input of κ, x, $m \in \{0,1\}^*$, run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^{p(\kappa)})$, $\mathsf{hk} \leftarrow \mathsf{SPHF.HashKG}(L_R)$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$ 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 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). To prove adaptive soundness we use a sequence of games.

Game 0: The original adaptive soundness game.

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

$\mathsf{Enc}(1^\kappa, x, m):$ On input of κ, x, $m \in \{0,1\}^*$, run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^{p(\kappa)})$, $\mathsf{hk} \leftarrow \mathsf{SPHF.Hash}$-$\mathsf{KG}(L_R)$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$ 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 property of the SPHF, the adversary's view in Game 1 is statistically close to the view in Game 0, i.e., Game 0 and Game 1 are statistically indistinguishable.

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

$\mathsf{Enc}(1^\kappa, x, m):$ On input of κ, x, $m \in \{0,1\}^*$, run $\mathsf{pp} \leftarrow \mathsf{SPHF.Setup}(1^{p(\kappa)})$, $\mathsf{hk} \leftarrow \mathsf{SPHF.Hash}$-$\mathsf{KG}(L_R)$, $\mathsf{hp} \leftarrow \mathsf{SPHF.ProjKG}(\mathsf{hk}, x)$. Then 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 $2^{(\ell_{\Sigma,\kappa} - |\mathsf{R}|)/2}$. This means that 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 $\mathsf{IND\text{-}T}$ 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 $\to$ Game 3: This is only a conceptual change.

The success probability of an adversary in Game 3 is bounded by the success probability in the IND-T game of the symmetric encryption scheme; a distinguisher between Game 0 and Game 3 would imply a distinguisher for statistically close distributions. $\square$

Remark 2. We note that we can use the same trick as [Zha14] to further reduce the ciphertext size of our constructions. That is, we can add an algorithm Setup, that takes a security parameter 1^κ, and run the setup of pp and H within Setup. Then, pp as well as H are treated as public parameters and are additionally provided to the algorithms Enc and Dec. This way, it is no longer required to generate them for every encryption and include them in every ciphertext.

4 Efficient Instantiations of SPHFs for Algebraic Languages

Recent expressive SPHFs are mostly constructed to be compatible with the universal composability (UC) framework [Can01]. Such constructions (e.g., [BBC$^+$13a]) usually build upon SPHFs based on IND-CCA2 secure (labeled) Cramer-Shoup encryption (yielding non-malleable and extractable commitments). Consequently, such constructions often trade maximum efficiency for UC compatibility. 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 that can be expressed via a set of pairing product equations (PPEs).

4.1 SPHF for Linear Encryptions

As a basis, we use the ElGamal-based SPHF by Gennaro and Lindell [GL03], 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 can in turn be decrypted by computing $M = {}^{e}/_{(u^{1/x_1} \cdot v^{1/x_2})}$. It is easy to show (as demonstrated by Boneh et al. in [BBS04]), that the scheme sketched above provides IND-CPA security 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 contains all triples $(\mathsf{pk}, C_M, M) \in \mathbb{G}^2 \times \mathbb{G}^3 \times \mathbb{G}$ of linear encryption public keys pk, ciphertexts C_M with respect to pk and corresponding messages $M \in \mathbb{G}$. Membership in this language is witnessed by the randomness $r = (r_1, r_2) \in \mathbb{Z}_p^2$ used to compute C_M.

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

$\mathsf{Setup}(1^\kappa):$ On input of κ, run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$ and return $\mathsf{pp} \leftarrow \mathsf{BG}$.
$\mathsf{HashKG}(L_R):$ On input of pp and L_R, return $\mathsf{hk} \leftarrow (\eta, \theta, \zeta) \xleftarrow{R} \mathbb{Z}_p^3$.
$\mathsf{ProjKG}(\mathsf{hk}, x):$ On input of pp, hk and some word $x = (\mathsf{pk}, C_M, M) \in \mathbb{G}^2 \times \mathbb{G}^3 \times \mathbb{G}$, where $\mathsf{pk} = (\mathsf{pk}_1, \mathsf{pk}_2) = (g^{x_1}, g^{x_2})$ and $C_M = (\mathsf{pk}_1^{r_1}, \mathsf{pk}_2^{r_2}, M \cdot g^{r_1+r_2})$, compute and return $\mathsf{hp} \leftarrow (\mathsf{hp}_1, \mathsf{hp}_2) = (\mathsf{pk}_1^\eta g^\zeta, \mathsf{pk}_2^\theta g^\zeta)$.
$\mathsf{Hash}(\mathsf{hk}, x):$ On input of pp, $\mathsf{hk} = (\eta, \theta, \zeta)$ 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 pp, 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

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 $\mathsf{hp} = (\mathsf{pk}_1^\eta g^\zeta, \mathsf{pk}_2^\theta g^\zeta)$, 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

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

It is easy to see that 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 (pk, C_M, M) is in fact in L_R. Conversely, if $(\mathsf{pk}, C_M, M) \notin L_R$, we have $r_3 \neq r_1 + r_2$ and the value H looks perfectly random. □

Proof (Pseudo-Randomness). We already know that smoothness holds and we now prove pseudo-randomness by showing that a distinguisher between the distributions considered in smoothness and pseudo-randomness is a distinguisher for DLIN. We obtain a DLIN instance $(\mathsf{BG}, g_1, g_2, g_1^r, g_2^s, g^t)$ and compute the ciphertext to Y_1 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. □

4.2 Extending Supported Languages

Applying techniques presented in [BBC+13a, BBC+13b] to the SPHF in Scheme 3 allows us to extend the set of supported languages to every kind of pairing product equations (PPEs).

In particular this allows us to construct SPHFs for languages defined for the satisfiability of PPEs of the form

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

where X_i, Y_i and $\mathbf{Z}_i$ remain secret and are encrypted using linear encryption. Since all elements are available in both groups, we can drop the second product in Equation 1 without loosing expressiveness and obtain:

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

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, $\mathsf{hk} = (\mathsf{hk}_i)_{i\in[n]}$, $\mathsf{hp} = (\mathsf{hp}_i)_{i\in[n]}$ and hashing as well as projective hashing are defined as follows.

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

Subsequently, we prove the following:

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

Subsequently, we show correctness, smoothness, and pseudo-randomness.

Proof (Correctness). For the sake of 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. Recall, that (r_{11}, r_{12}) and (r_{21}, r_{22}) represent the witness. Then, the projective hash value using the projection key hp is computed as

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

Computing the hash value using the hashing key hk yields:

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

where for the last step (ii), we use the fact that $B = e(A_1, Y_1) \cdot \mathbf{Z}_2^{\gamma_2}$ 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 PPEis not satisfied. As Y_i and $\mathbf{Z}_i$ cancel out via multiplication by $B^{-\zeta}$ when plugging in the commitments into the PPE, we know that—by the smoothness of the underlying SPHF—for the hash value we have that $\log_g H \notin \text{span}(\log_g \mathsf{hp}_1, \log_g \mathsf{hp}_2)$. Consequently, 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. □

4.3 SPHF for Linear Groth-Sahai Commitments

Now we define the language for an SPHF by the commitments which were already used in a GS proof for the satisfiability of some PPE. This brings us closer to our application, 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. In general, we work with row vectors and use the binary operators $\circ : \mathbb{G}^{1\times n} \times \mathbb{G}^{1\times n} \to \mathbb{G}^{1\times n}$ and $\cdot : \mathbb{Z}_p \times \mathbb{G}^{1\times n} \to \mathbb{G}^{1\times n}$. Here, $\circ$ denotes entry-wise multiplications, whereas $\cdot$ denotes entry-wise exponentiation.

Linear GS Commitments. First, let us 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 $\rho, \nu \xleftarrow{R} \mathbb{Z}_p$. Here, the CRS is setup to yield perfectly binding commitments. In contrast, in the perfectly hiding setup we have that $\log_g U_3 \notin \text{span}(\log_g U_1, \log_g U_2)$. The two setups are computationally indistinguishable under DLIN. Thus, we can align our further explanations to the perfectly binding setup and they equally apply to the perfectly hiding case. A commitment to a message $M \in \mathbb{G}$ is computed by choosing $r_1, r_2, r_3 \xleftarrow{R} \mathbb{Z}_p$ and computing

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

Note that a commitment in this setting is a linear encryption of M with respect to randomness $((r_1 + \rho r_3), (r_2 + \nu r_3))$.

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). Now, we investigate the security of the SPHF in Scheme 4.

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

$\mathsf{Setup}(1^\kappa)$: On input of κ, run $\mathsf{BG} \leftarrow \mathsf{BGGen}(1^\kappa)$ and return $\mathsf{pp} \leftarrow \mathsf{BG}$.
$\mathsf{HashKG}(L_R)$: On input of pp and L_R, return $\mathsf{hk} \leftarrow (\eta, \theta, \zeta) \xleftarrow{R} \mathbb{Z}_p^3$.
$\mathsf{ProjKG}(\mathsf{hk}, x)$: On input of pp, hk and some word $x = (\mathsf{pk}, C_M, M) \in \mathbb{G}^6 \times \mathbb{G}^3 \times \mathbb{G}$, where $\mathsf{pk} = (g_1, g_2, g, g_1^\rho, g_2^\nu, g^{\rho+\nu})$ and $C_M = (g_1^{r_1} g_1^{\rho r_3}, g_2^{r_2} g_2^{\nu r_3}, M \cdot g^{r_1+r_2+r_3(\rho+\nu)})$, compute and return $\mathsf{hp} \leftarrow (\mathsf{hp}_1, \mathsf{hp}_2, \mathsf{hp}_3) = (g_1^\eta g^\zeta, g_2^\theta g^\zeta, (g_1^\rho)^\eta (g_2^\nu)^\theta (g^{\rho+\nu})^\zeta)$.
$\mathsf{Hash}(\mathsf{hk}, x)$: On input of pp, $\mathsf{hk} = (\eta, \theta, \zeta)$ and $x = (\mathsf{pk}, C_M, M) \in \mathbb{G}^6 \times \mathbb{G}^3 \times \mathbb{G}$, 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 pp, 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 (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

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

which proves correctness. □

Proof (Pseudo-Randomness). 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 $\mathsf{hp} = (g_1^\eta g^\zeta, g_2^\theta g^\zeta, g_1^{\rho\eta} g_2^{\nu\theta} g^{(\rho+\nu)\zeta})$, 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

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

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

Proof (Smoothness). We know that smoothness holds. A distinguisher between the distributions in smoothness and pseudorandomness distinguishes valid from invalid commitments and is, thus, a DLIN distinguisher. □

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, $\mathsf{hk} = (\mathsf{hk}_i)_{i\in[n]}$, $\mathsf{hp} = (\mathsf{hp}_i)_{i\in[n]}$ and we define

$$H_{\mathsf{Hash}} := B^{-\zeta} \cdot \prod_{i=1}^{m} e(A_i, u_i^{\eta_i} v_i^{\theta_i} e_i^{\zeta}) \cdot \prod_{i=m+1}^{n} (\mathbf{u}_i^{\eta_i}\mathbf{v}_i^{\theta_i}\mathbf{e}_i^{\zeta})^{\gamma_i} = \prod_{i=1}^{m} e(A_i, \mathsf{hp}_1^{r_1}\cdot \mathsf{hp}_2^{r_2}\cdot \mathsf{hp}_3^{r_3}) \cdot \prod_{i=m+1}^{n} e(g^{\gamma_i}, \mathsf{hp}_1^{r_1}\cdot \mathsf{hp}_2^{r_2}\cdot \mathsf{hp}_3^{r_3}) =: H_{\mathsf{Proj}}. \quad (3)$$

Security, of the construction in Equation (3) is easy to verify by the security of Scheme 4 using the same argumentation as in Section 4.2. 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* (2)*.*

4.5 Encrypting With Respect to a GS Proof

Plugging the SPHF in Scheme 4, used as demonstrated in Equation (3), into our generic WE constructions finally yields a methodology to encrypt a message m with respect to a large class of statements over PPEs proven using the GS framework.

To give a brief overview of our idea, assume that a prover creates a proof π for the satisfiability of some PPE. Informally, a GS proof boils down to sending 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 compute an encryption of 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 $\mathsf{GS} = (\mathsf{BGGen}, \mathsf{CRSGen}, \mathsf{Proof}, \mathsf{Verify})$ 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 provided as input to both, the encryptor E and the decryptor D. For simplicity our illustration is based on PPEs of the form in Equation (2), but we emphasize that our approach works for all statements expressible by PPEs of the form in Equation (1).

$$\begin{array}{lcr}
\mathsf{D}(\mathsf{BG}, \mathsf{crs}) & & \mathsf{E}(\mathsf{BG}, \mathsf{crs}) \\
\hline
\pi \leftarrow \mathsf{GS.Proof}(\mathsf{BG}, \mathsf{crs}, \mathsf{PPE}, (Y_i)_{i\in[n]}; r) & & \\
\text{store } r & \xrightarrow{\pi} & \\
 & & \text{parse } \pi \text{ as } ((C_i)_{i\in[n]}, \mathsf{PPE}, \pi_{\mathsf{GS}}) \\
 & \xleftarrow{c} & c \leftarrow \mathsf{WE.Enc}(\mathsf{BG}, ((C_i)_{i\in[n]}, \mathsf{PPE}), m) \\
\text{extract } r_{\mathsf{com}} \text{ from } r & & \\
m \leftarrow \mathsf{WE.Dec}(c, r_{\mathsf{com}}) & &
\end{array}$$

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

Observe that we require a slight conceptual change of the WE scheme. In particular, given a GS proof for some statement, the bilinear group is already fixed prior to calling WE.Enc.

This means that the encryption algorithm of the WE scheme additionally takes the bilinear group description BG and uses it as pp (instead of internally generating it). Recall Remark 2, that this will even reduce the size of the ciphertexts. For notational convenience, let us view a GS proof π as a sequence of commitments $(C_i)_{i\in[n]}$, a corresponding PPE and a proof part π_{GS}. Additionally, we make the randomness r used in $\mathsf{GS.Proof}$ explicit and assume that one can efficiently extract the randomness r_{com} used in the commitments $(C_i)_{i\in[n]}$ in π from the entire randomness r.

Discussion. It is easy to see that E only operates on public information, which means that our construction does not influence the security of the underlying GS proof system. Furthermore, the message m gets encrypted with respect to a proof π. The statement includes the PPE as well as the commitments to the unrevealed values $(Y_i)_{i\in[n]}$ in the PPE, whereas the randomness r_{com} serves as witness (i.e., as decryption key). It remains to discuss why it is it interesting to use the randomness as "decryption key". Observe that giving away the randomness destroys the witness indistinguishability/zero-knowledge property of the GS proof, meaning that the prover will have a strong interest in keeping r_{com} and thus the "decryption key" secret.

5 How to Reply an Unknown Whistleblower

Finally, we come back to our initial question. Recall, that we assumed that a whistleblower Edwarda can leak authoritative secrets in a way that the public will be convinced of its authenticity, but she wants to stay anonymous. She signs the 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. And we asked ourselves if a journalist can confindentially reply the unknown whistleblower.

Now we positively answer the question and show how to use WE and the techniques from Section 4.5 together with ring signatures to allow anyone to send a confidential reply to the whistleblower. In particular, we show how 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).

We start by recalling a generic construction of ring signatures for bilinear groups [Gha13] obtained from $\mathsf{EUF\text{-}CMA}$-secure signature schemes and witness-indistinguishable GS proofs in Section 5.1. Then, in Section 5.2 we discuss how to generically extend this construction with the feature to allow for confidential replies to the unknown (anonymous) signer and provide a brief performance analysis in Section 5.3.

5.1 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 as well as one instantiation subsequently, where we adapt the notation to ours. Here, $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is 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 $\mathsf{Sig.Setup}$ that outputs a bilinear group description BG and all other algorithms take BG as an additional input. Furthermore, $\mathsf{GS} = (\mathsf{BGGen}, \mathsf{CRSGen}, \mathsf{Proof}, \mathsf{Verify})$ denotes the non-interactive GS proof system. Henceforth, let the output of $\mathsf{Sig.Setup}$ and $\mathsf{GS.BGGen}$ be compatible, i.e., the parameters 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, let $F : \{0,1\}^* \rightarrow \mathcal{M}$ be a collision resistant hash function.

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

We note that 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.

$\mathsf{Setup}(1^\kappa)$: On input of κ, run $\mathsf{BG} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$, $\mathsf{crs} \leftarrow \mathsf{GS.CRSGen}(\mathsf{BG})$, choose a collision resistant hash function $F : \{0,1\}^* \rightarrow \mathcal{M}$ 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 3 ([Gha13]). *The generic construction in Scheme 6 is secure, if the underlying signature scheme is* EUF-CMA *secure, F is a collision resistant hash function, and* GS *is sound and witness indistinguishable.*

5.2 Encrypting a Confidential Reply to an Unknown Receiver

For every ring signature scheme following the generic construction 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}. In Appendix B we provide a concrete instantiation of Scheme 6 using Waters signatures [Wat05] as an example of a ring signature scheme that is compatible with our framework.

What comes in handy is that revealing the randomness used in the commitments in π_{L_1}—and, thus, the required information to decrypt (as already discussed in Section 4.5)—in this application implies revealing the signer's identity. This, however, conflicts with the signer's interest to remain anonymous and thus discourages the signer to do so. Finally, we note that the entity replying to the whistleblower might again use ring signatures to ensure the authenticity of the encrypted reply, which, in turn, opens a channel for another confidential reply.

5.3 Efficiency of our Approach

We want to emphasize that our approach 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 $\mathsf{ProjHash}$, respectively. The operations required when using this SPHF for languages over bilinear groups as demonstrated in Equation (3), 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 size of the PPE (in particular in n and m, respectively) and is almost as efficient as evaluating the PPE with plain values.

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

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

To finally underline the practical relevance of our construction with respect to our application scenario, 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*. Just to give an intuition of how much this will cost, we assume a Type-3 setting and use performance values of a BN-pairing implementation on an ARM Cortex-M0+ with a drop in hardware accelerator [UW14], where 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 can be performed in approximately 500ms. We stress that encryption/decryption will most likely be performed on more powerful devices such as desktop PCs. In this case, the computational overhead will be in the order of milliseconds and thus not even yield a noticeable delay [Mit13].

6 Discussion

Finally, we informally discuss two potential applications of the methodologies presented in this paper and leave their rigorous investigation as future work.

Ring Encryption. It is quite straightforward to construct a ring encryption equivalent to group encryption [KTY07], i.e., and *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 proofs), 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 to the one in [Gha13]). Our use of WE in Section 5 can be interpreted as a construction for a variation of *ring encryption* in the sense that we additionally achieve *unconditionally 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 (cf. Section 4.5) or in particular with respect to a ring signatures (cf. Section 5) could be applied to LAKE. More precisely, two parties that do not want to reveal their identity to each other (but only the respective membership to potentially distinct and independently chosen ad-hoc rings) can agree on a common secret key, i.e., by using an SPHF for languages covering ring signatures as demonstrated in Section 5.

References

[AFP15] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline Witness Encryption. *IACR Cryptology ePrint Archive*, 2015:838, 2015.

[AGH15] Joseph A. Akinyele, Christina Garman, and Susan Hohenberger. Automating Fast and Secure Translations from Type-I to Type-III Pairing Schemes. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-6, 2015*, pages 1370–1381, 2015.

[BBC$^+$13a] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. Efficient UC-Secure Authenticated Key-Exchange for Algebraic Languages. In *PKC*, volume 7778 of *LNCS*, pages 272–291. Springer, 2013.

[BBC$^+$13b] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New Techniques for SPHFs and Efficient One-Round PAKE Protocols. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 449–475, 2013.

[BBDP01] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-Privacy in Public-Key Encryption. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, pages 566–582, 2001.

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

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

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

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

[BJSW14] Joshua Brody, Sune K. Jakobsen, Dominik Scheder, and Peter Winkler. Cryptogenography. In *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 13–22, 2014.

[BKM09] Adam Bender, Jonathan Katz, and Ruggero Morselli. Ring Signatures: Stronger Definitions, and Constructions without Random Oracles. *J. Cryptology*, 22(1):114–138, 2009.

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

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

[CHKM10] Sanjit Chatterjee, Darrel Hankerson, Edward Knapp, and Alfred Menezes. Comparing two Pairing-Based Aggregate Signature Schemes. *Des. Codes Cryptography*, 55(2-3):141–167, 2010.

[COR99] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional Oblivious Transfer and Timed-Release Encryption. In *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, pages 74–89, 1999.

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

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

[FNV15] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. Cryptology ePrint Archive, Report 2015/740, 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 *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49, 2013.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the Implausibility of Differing-Inputs Obfuscation and Extractable Witness Encryption with Auxiliary Input. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, pages 518–535, 2014.

[GGSW13a] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness Encryption and its Applications. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 467–476, 2013.

[GGSW13b] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness Encryption and its Applications. Cryptology ePrint Archive, Report 2013/258, 2013.

[Gha13] Essam Ghadafi. Sub-linear Blind Ring Signatures without Random Oracles. In *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, pages 304–323, 2013.

[GKP^{+}13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to Run Turing Machines on Encrypted Data. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 536–553, 2013.

[GL03] Rosario Gennaro and Yehuda Lindell. A Framework for Password-Based Authenticated Key Exchange. In *Advances in Cryptology - EUROCRYPT 2003*, pages 524–543, 2003.

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

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *J. Comput. Syst. Sci.*, 28(2):270–299, 1984.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.

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

[GS08] Jens Groth and Amit Sahai. Efficient Non-interactive Proof Systems for Bilinear Groups. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 415–432, 2008.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[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*, 2015:478, 2015.

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

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

[JO16] Sune K. Jakobsen and Claudio Orlandi. How to Bootstrap Anonymous Communication. In *Innovations in Theoretical Computer Science, ITCS'16, Cambridge, Massachusetts, USA, January 14-16, 2016*, 2016.

[KD04] Kaoru Kurosawa and Yvo Desmedt. A New Paradigm of Hybrid Encryption Scheme. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 426–442, 2004.

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

[KPSY09] Eike Kiltz, Krzysztof Pietrzak, Martijn Stam, and Moti Yung. A New Randomness Extraction Paradigm for Hybrid Encryption. In *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 590–609, 2009.

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

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

[Mit13] Shigeo Mitsunari. A fast implementation of the optimal ate pairing over bn curve on intel haswell processor. Cryptology ePrint Archive, Report 2013/362, 2013.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to Leak a Secret. In *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security*, pages 552–565, 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 *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473, 2005.

[UW14] Thomas Unterluggauer and Erich Wenger. Efficient Pairings and ECC for Embedded Systems. In *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, pages 298–315, 2014.

[Wat05] Brent Waters. Efficient Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 114–127, 2005.

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

[Zha14] Mark Zhandry. How to Avoid Obfuscation Using Witness PRFs. *IACR Cryptology ePrint Archive*, 2014:301, 2014.

A Digital Signatures

Followingly, we recall the formal definition of digital signature schemes.

Definition 17 (Digital Signatures). *A digital signature scheme* DSS *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* $\kappa \in \mathbb{N}$ *as input and outputs a secret (signing) key* sk *and a public (verification) key* pk *with associated message space* $\mathcal{M}$ *(we omit to mention the message space* $\mathcal{M}$ *and assume that it is implicit in the public key).*

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

$\mathsf{Verify}(\mathsf{pk}, m, \sigma)$: *The deterministic verification algorithm, which takes a signature* σ*, a message* $m \in \mathcal{M}$ *and a public key* pk *as input, and outputs* 1 *if* σ *is a valid signature for* m *under* pk *or* 0 *otherwise.*

For security digital signature schemes are required to be correct and existentially unforgeable under chosen message attacks ($\mathsf{EUF\text{-}CMA}$ secure [GMR88]). Formal definitions are not required in the context of this paper and thus omitted.

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 based on Waters signatures [Wat05] as presented in [Gha13] and recalled in Scheme 7 (here 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 $\mathsf{BG} \leftarrow \mathsf{Sig.Setup}(1^\lambda)$, $\mathsf{crs} \leftarrow \mathsf{GS.CRSGen}(\mathsf{BG})$ 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.

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

Scheme 7: Waters signature based ring signature scheme [Gha13]

Theorem 4 ([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.*

We note that Waters signatures can also straightforwardly be instantiated in a Type-3 setting [CHKM10].