Identity-Based Encryption in DDH Hard Groups

Olivier Blazy, Saqib Kakvi

Identity Based Encryption in DDH hard Groups

Olivier Blazy[1][0000−0001−6205−8249] and Saqib A. Kakvi[2][0000−0003−4425−4240]

[1] École Polytechnique, France
olivier.blazy@polytechnique.edu
[2] Royal Holloway University of London, United Kingdom
kakvi@rhul.ac.uk

Abstract. The concept of Identity-Based Encryption was first introduced by Shamir (CRYPTO 1984) but were not realised until much later by Sakai, Ohgishi and Kasahara (SCIS 2000), Boneh and Franklin (CRYPTO 2001) and Cocks (IMACC 2001). Since then, Identity-Based Encryption has been a highly active area of research. While there have been several instantiations of Identity-Based Encryption and its variants, there is one glaring omission: there have been no instantiations in plain Decisional Diffie-Hellman groups. This seemed at odds with the fact that we can instantiate almost every single cryptographic primitive in plain Decisional Diffie-Hellman groups. An answer to this question came in a result by Papakonstantinou, Rackoff and Vahlis (EPRINT 2012), who showed that it is *impossible* to instantiate an Identity-Based Encryption in plain DDH groups. The impossibility result was questioned when Döttling and Garg (CRYPTO 2017) presented an Identity-Based Encryption based on the Decisional Diffie-Hellman problem. This result however did not disprove the impossibility result, as it requires the use of garbled circuits, which are inherently interactive. This type of scheme is not covered by the impossibility result, but it does raise some questions. In this paper, we answer some of those questions by constructing an Identity-Based Encryption scheme based on the Decisional Diffie-Hellman problem. We achieve this by instantiating the generic construction based on Witness Encryption by Garg, Gentry, Sahai and Waters (STOC 2013), with some minor changes. To this end, we construct the first unique signature scheme in Decisional Diffie-Hellman groups, to the best of our knowledge. The unique signature scheme, and as a result, our Identity-Based Encryption scheme, is inefficient, but this is unavoidable. Our construction does not completely contradict the impossibility result, but instead shows that the statement was too strong, and the result only rules out efficient constructions.
Keywords: Identity-Based encryption, Unique Signatures, Generic Constructions, DDH, Impossibility results

1 Introduction

Identity-Based Encryption (IBE) was first proposed by Shamir [43] is a generalization of standard Public Key Encryption (PKE), wherein instead of each user generating a public key themselves, their unique identifier, such as their

e-mail address, would serve as their public key. The corresponding secret key for decryption is derived from the public identity and a master secret key, which is held by a trusted third party. Despite being posited in the 80s, the first constructions came at the turn of the millennium with a factoring-based scheme due to Cocks [16] and the pairing-based schemes due to Boneh and Franklin [10] and Sakai, Ohgishi and Kasahara [40]. A few years later, an IBE scheme based on lattice problems was introduced by Gentry, Peikert and Vaikuntanathan [26]. Since then, IBE schemes have been an active area of research, with several interesting results.

Despite all the results and advances in the field of IBEs, there is one glaring omission; there is no IBE scheme that is secure in plain Decisional Diffie-Hellman (DDH) groups. Given this, one might think that parings are somehow crucial for the constructions of IBEs and that it may not be possible to construct IBEs in plain DDH groups. Further evidence to this end was provided by the impossibility result of Papakonstantinou, Rackoff, and Vahlis [37], who showed that it is impossible to construct a weakly secure IBE scheme in the Generic Group Model (GGM). We will refer to this as the PRV impossibility result henceforth. Given that this weakly secure scheme cannot exist in this idealised model, it follows that no fully secure scheme can exist in this idealised model. This cast further doubt upon the possibility of such a scheme being realised in the standard model. However, recent developments present a glimmer of hope and may show a way in which to construct an IBE in DDH groups.

The seminal work of Garg, Gentry, Sahai and Waters [23] not only introduced the very powerful primitive of Witness Encryption (WE), but they also showed how to use it to generically construct several other primitives. In particular Garg et al. showed how to *generically* construct a weakly secure IBE scheme, from a WE scheme and a unique signature scheme. If we assume that both the generic construction and the impossibility result are correct, this would mean that at least one of the constituent components cannot be instantiated. However, if we can show an instantiation of these components, then either the impossibility result is correct and the DDH problem is not hard, or the impossibility result is somehow flawed. Given this, that means that at least one of the following statements is true:

1. There is no unique signature scheme in DDH groups.
2. There is no secure WE in DDH groups.
3. The DDH problem is not hard.
4. The PRV impossibility result is incorrect.

We now take this list and examine each statement and begin to eliminate the incorrect from the list. Once we have removed all the incorrect statements, we will know where we stand. We can show that the first two statements are not true by instantiating a WE scheme and a unique signature scheme in DDH groups. Once we have these, we can then apply the transformation to them to construct an IBE scheme. Given this, we can then look at the third and fourth items on our list.

We begin by showing a unique signature scheme in DDH groups. To the best of our knowledge, there are no deterministic signatures in plain DDH groups. The most commonly used Diffie-Hellman-based signatures are variants of the Digital Signature Algorithm (DSA) [42], which requires randomness. DSA builds upon the early work of ElGamal [22] and Schnorr [41], both of which are randomised. Even the tightly secure standard model signatures of Blazy et al. [8] rely on the randomness of Chameleon Hash Functions to be secure. This is not surprising since it was shown that any deterministic signature in the standard model cannot have a tight security reduction [17, 31]. Indeed, not only can signatures in DDH groups not be tight, but the can also not be short as shown by Döttling et al. [21]. This means that we can do no better than a non-tight, non-short signatures. Keeping this in mind we construct a unique signature scheme by combining the discrete logarithm-based hash function of Chaum, van Heijst and Pfitzmann [13] with the tree-based signature construction of Merkle [35] in Section 3.1.

The next item on the list is the WE, which at first glance seems like a daunting task as all known constructions of WE (and variants thereof) [2, 12, 15, 23] rely on either indistinguishability Obfuscation (iO) [5] or extractable Obfuscation (eO) [12], neither of which have an instantiation in DDH groups. While it might seem that we have hit a wall, we are saved by another primitive, namely Smooth Projective Hash Functions (SPHFs), which were first introduced by Cramer and Shoup [19]. There are several known constructions of SPHFs in DDH groups [19, 24], for a variety of languages. Furthermore, it has been shown that an SPHF for some language $\mathcal{L}$ can be used to construct a WE scheme for $\mathcal{L}$ [1, 7]. We note that this generic transformation does not give rise to a generic WE scheme, but rather a WE for the given language $\mathcal{L}$. We describe the language and the related SPHF in more detail in Section 3.2. We then show how to build a WE scheme from our SPHF. Therefore we know that the second item on our list is also not true.

Given that we have a unique signature and a WE in DDH groups, we also have an IBE in DDH groups. At this point, we must note that our scheme is a proof of concept and is inefficient. Although the scheme could be optimized to some degree, this inefficiency is unavoidable. This is due to the fact that our unique signature scheme cannot be efficient, as was demonstrated by Döttling et al. [21]. However, this will be a key point in showing the limits of the PVR result.

We now look at the third and fourth items on our list simultaneously. Our IBE construction and the PRV impossibility result are in direct contradiction of one another. This means that either the PRV impossibility result is correct and our IBE construction gives an adversary against the DDH problem, which would mean that the DDH problem is not hard. The other possibility is that the the DDH problem is still hard, which would mean that the PRV impossibility result is not correct. Here we note that the PVR result and the result of Döttling et al. [21] have a similar high-level idea; they show that with sufficiently many queries, one can recover the secret key. Therefore, we see that the PVR impossibility result

is incorrect and the statement they show is somewhat weaker. Specifically, the PVR result only rules out *efficient* IBEs. We discuss this in Section 4.

1.1 Our Contributions

The core result of the paper is an IBE in DDH groups. To build this, we proceed in the following steps:

- Firstly, we need to build a unique signature scheme secure under the DDH assumption. The first step is to build a one-time signature. We achieve this by treating the hash function of Chaum, van Heijst and Pfitzmann [13] as Chameleon Hash Function [33] and applying the techniques of Mohassel [36].
- The next step in constructing our signatures is amplifying our one-time signature to a k-time signature for sufficiently large k. We achieve this using a variant of Merkle's tree-based signatures [35]. To be in a purely DDH setting, we rely again on the hash function by Chaum, van Heijst and Pfitzmann [13].
- Secondly, we construct a witness encryption using SPHFs. The original generic construction of Garg et al. [23] uses the Goldreich-Levin hardcore predicate [27], but we cannot use this in the GGM, as we are required to do XORs and the SPHFs for this type of circuit require interactivity. To work around this we replace the Goldreich-Levin hardcore predicate with a Katz-Wang style DDH proof of membership [32], for which we can build efficient SPHFs. It is clear to see that our replacement satisfies the original generic construction, at the cost of being specific to DDH hard groups. We then use the well-known WE construction from an SPHF.
- Finally, we combine these two using our modified version of the generic construction of Garg et al. [23] to build a selectively secure IBE in the GGM.

Once we have our IBE scheme, this gives us a first indication that the PVR impossibility result is incorrect. It is not a direct contradiction as our construction is not tight and has large keys and ciphertexts. However, as we discuss in Section 4, this shows that the result only extends to *efficient* instantiations.

1.2 Related Work

Identity-Based Encryption schemes have been widely studied since their introduction by Shamir in [43], while the design of IBEs has been studied extensively [10,45], there is a lack of generic design which could be leveraged to achieve a construction natively under a given hypothesis. Chen and Wee [14] and Blazy, Kiltz and Pan [9] proposed a generic design of IBEs by using affine MACs as a building block. However their constructions, while leading to (almost) tight IBEs, require the use of pairings, which puts them outside pure DDH groups. More recently, Döttling and Garg [20] proposed a construction of IBE under DDH using Garbled Circuits and Pedersen-like multi-commitments. Their construction was the first to try and bridge the gap. However, Garbled Circuit while being a useful building blocks are inherently interactive, which does not match

with the classical definitions of IBE. In the standard definition, neither the encryptors nor the recipients are expected to actively interact with the authority to produce/decrypt a new ciphertext. Thus, this IBE is not ruled out by the PVR impossibility result. Our construction is similar to that of Döttling and Garg [20], but we use WE instead of garbled circuits and thus do not require interactivity.

2 Preliminaries

2.1 Notations and conventions

We denote our security parameter as λ. For all $n \in \mathbb{N}$, we denote by 1^n the n-bit string of all ones. For any bit string a, we define $a[i]$ as the i^{th} most significant bit of a. For any element x in a set S, we use $x \in_R S$ to indicate that we choose x uniformly random in S. We denote the set of all integers with $\mathbb{Z}$ and the set of all primes as $\mathbb{P}$. We denote the set of k-bit integers as $\mathbb{Z}[k]$ and the set of all k-bit primes as $\mathbb{P}[k]$. All algorithms may be randomized. For any algorithm A, we define $x \xleftarrow{\$} A(a_1, \ldots, a_n)$ as the execution of A with inputs $a_1, \ldots, a_n$ and fresh randomness and then assigning the output to x. For conciseness, we will write PPT for Probabilistic Polynomial Time.

2.2 Identity-Based Key Encryption

We recall syntax of an Identity-Based Encryption scheme.

Definition 1 (Identity-Based Key Encryption). *An Identity-Based Encryption $\mathcal{IBE}$ consists of four PPT algorithms $\mathcal{IBE} = (\mathcal{IBE}.\mathsf{Setup}, \mathcal{IBE}.\mathsf{KeyGen}, \mathcal{IBE}.\mathsf{Encrypt}, \mathcal{IBE}.\mathsf{Decrypt})$ with the following properties.*

- *The probabilistic key generation algorithm $\mathcal{IBE}.\mathsf{Setup}(1^\lambda)$ returns the (master) public/secret keys $(\mathsf{pk}, \mathsf{msk})$.*
- *The probabilistic user secret key generation algorithm $\mathcal{IBE}.\mathsf{KeyGen}(\mathsf{msk}, \mathtt{ID})$ returns a secret key $\mathsf{sk}_{\mathtt{ID}}$ for identity $\mathtt{ID} \in \mathfrak{I}$.*
- *The probabilistic encapsulation algorithm $\mathcal{IBE}.\mathsf{Encrypt}(\mathsf{pk}, \mathtt{ID}, m)$ returns a ciphertext c with respect to the $\mathtt{ID} \in \mathfrak{I}$.*
- *The deterministic decryption algorithm $\mathcal{IBE}.\mathsf{Decrypt}(\mathsf{sk}_{\mathtt{ID}}, \mathtt{ID}, c)$ returns a plaintext m or the reject symbol $\perp$.*

For correctness we require that for all $\lambda \in \mathbb{N}$, all pairs $(\mathsf{pk}, \mathsf{msk})$ generated by $\mathcal{IBE}.\mathsf{Setup}(1^\lambda)$, all $\mathtt{ID} \in \mathfrak{I}$, all $\mathsf{sk}_{\mathtt{ID}}$ generated by $\mathcal{IBE}.\mathsf{KeyGen}(\mathsf{msk}, \mathtt{ID})$ and all (c) generated by $\mathcal{IBE}.\mathsf{Encrypt}(\mathsf{pk}, \mathtt{ID}, m)$ for all m:

$$\Pr[\mathcal{IBE}.\mathsf{Decrypt}(\mathsf{sk}_{\mathtt{ID}}, \mathtt{ID}, c) = m] = 1.$$

2.3 Signature Schemes

We first recall the definition of a unique signature scheme.

Definition 2. *A digital signature scheme $\mathcal{SIG}$ with message space $\mathfrak{M}$ and signature space $\mathfrak{S}$ is defined as a triple of PPT algorithms $\mathcal{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:*

- KeyGen *takes as an input the unary representation of our security parameter 1^λ and outputs a signing key sk and verification key pk.*
- Sign *takes as input a signing key sk, message $m \in \mathfrak{M}$ and outputs a signature $\sigma \in \mathfrak{S}$.*
- Verify *is a deterministic algorithm, which on input of a public key and a message-signature pair $(m, \sigma) \in \mathfrak{M} \times \mathfrak{S}$ outputs 1 (accept) or 0 (reject).*

We say $\mathcal{SIG}$ is correct if for any $\lambda \in \mathbb{N}$, all $(\mathsf{pk}, \mathsf{sk}) \leftarrow_s \mathsf{KeyGen}(1^\lambda)$, all $m \in \mathfrak{M}$, and all $\sigma \leftarrow_s \mathsf{Sign}(\mathsf{sk}, m)$ we have that

$$\Pr[\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1] = 1.$$

We say $\mathcal{SIG}$ is a unique signature scheme if $\forall m \in \mathfrak{M}, \exists!\ \sigma \in \mathfrak{S}$ such that $\mathsf{Verify}(\mathsf{pk}, m, \sigma) = 1$.

2.4 Witness Encryption

Witness Encryption (WE), which is generalisation of public key encryption where anybody in possession of a valid witness w that some statement x is in a specified language can decrypt all ciphertexts encrypted under x. The first witness encryption scheme was presented by Garg et al. [23]. We now recall the definition of WE.

Definition 3. *A Witness Encryption scheme $\mathcal{WE}$ for some language $\mathcal{L}$ is defined by the two PPT algorithms $\mathcal{WE} = (\mathsf{Encrypt}, \mathsf{Decrypt})$:*

- $\mathsf{Encrypt}$ *takes as an input the unary representation of our security parameter 1^λ, an instance x and a message m and outputs a ciphertext c.*
- $\mathsf{Decrypt}$ *takes as input the decryption parameters pp_d, a ciphertext c and a witness w and outputs m if $(x, w) \in \mathcal{L}$ and $\perp$ otherwise.*

We say $\mathcal{WE}$ is correct if for all messages m and for all pairs $(x, w) \in \mathcal{L}$, we have:

$$\Pr[\mathsf{Decrypt}(\mathsf{Encrypt}(1^\lambda, x, m), w) = m] = 1$$

2.5 Smooth Projective Hash Functions

When building our Witness Encryption, we require Smooth Projective Hash Functions (SPHF). They were proposed by Cramer and Shoup in [19]. SPHFs can evaluated them in two ways : using the (secret) hashing key on a public element, or using the (public) projected key on a word on a special subset of its domain with a secret witness.

A Smooth Projective Hash Function system over a language $\mathcal{L} \subset X$ onto a set $\mathcal{S}$ is defined by the five following algorithms :

– $\mathsf{Setup}(1^\lambda)$ generates the global parameters of the protocol and the description of an $\mathcal{NP}$ language $\mathcal{L}$.
– $\mathsf{HashKG}(\mathcal{L})$ generates a hashing key hk.
– $\mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, C)$ derives the projection key hp, possibly depending on the word C.
– $\mathsf{Hash}(\mathsf{hk}, \mathcal{L}, C)$ outputs the hash value of C from hk.
– $\mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, C, w)$ outputs the hash value of the word C from the projection key hp and the witness w that $C \in \mathcal{L}$.

A Smooth Projective Hash Function SPHF should satisfy the following properties:

– *Correctness*: Let $W \in \mathcal{L}$ and w a witness of this membership. Then, for all hashing keys hk and associated projection keys hp we have $\mathsf{Hash}(\mathsf{hk}\mathcal{L}, W) = \mathsf{ProjHash}(\mathsf{hp}, \mathcal{L}, W, w)$.
– *Smoothness*: For all $W \in X \setminus \mathcal{L}$ the following distributions are statistically indistinguishable:

$$\Delta_0 = \left\{ (\mathcal{L}, \mathsf{prm}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{prm} = \mathsf{Setup}(1^\lambda), \mathsf{hk} = \mathsf{HashKG}(\mathcal{L}), \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, W), v = \mathsf{Hash}(\mathsf{hk}, \mathcal{L}, W) \end{array} \right\}$$

$$\Delta_1 = \left\{ (\mathcal{L}, \mathsf{prm}, W, \mathsf{hp}, v) \;\middle|\; \begin{array}{l} \mathsf{prm} = \mathsf{Setup}(1^\lambda), \mathsf{hk} = \mathsf{HashKG}\mathcal{L}, \\ \mathsf{hp} = \mathsf{ProjKG}(\mathsf{hk}, \mathcal{L}, W), v \xleftarrow{\$} \mathcal{S} \end{array} \right\}.$$

This is formalized by

$$\mathsf{Adv}^{\mathsf{StdM}}_{\mathsf{SPHF}}(1^\lambda) = \sum_{V \in \mathbb{G}} \left| \Pr_{\Delta_1}[v = V] - \Pr_{\Delta_0}[v = V] \right| \text{ is negligible.}$$

– *Pseudo-Randomness*: If $W \in \mathcal{L}$, then without a witness of membership the two previous distributions should remain computationally indistinguishable: for any adversary $\mathcal{A}$ within reasonable time and a negligible function ϵ, we have

$$\begin{aligned} Adv^{\mathsf{pr}}_{\mathsf{SPHF},\mathcal{A}}(1^\lambda) &= |\Pr_{\Delta_1}[\mathcal{A}(\mathcal{L}, \mathsf{prm}, W, \mathsf{hp}, v) = 1] - \Pr_{\Delta_0}[\mathcal{A}(\mathcal{L}, \mathsf{prm}, W, \mathsf{hp}, v) = 1]| \\ &\leqslant \epsilon \end{aligned}$$

2.6 The Generic Group Model

The Generic Group Model was first proposed by Shoup [44], as a way to show lower bounds for *generic* algorithms solving the discrete logarithm problem, such as Pollard's Rho method [39]. Shoup presented a model wherein an adversary only interacts with group elements via oracles for all the operations. To facilitate this, the adversary was given a random encoding as a unique tag for each element. The adversary can then use these encodings to make oracles queries on the

relevant elements. This model was then extended by Maurer [34], by replacing the random encodings with indices of their locations in the oracle's internal table.

One may wonder as to which model is better suited for use in cryptographic proofs. The answer to this was provided by Jager and Shwenk [28], who show the two models to be equivalent. They achieve this by reducing algorithms in model to the other. As the PVR impossibility result was stated in Shoup's model, we will also use Shoup's model for consistency.

Definition 4 (Generic Group Algorithm in Shoup's Model [44]). *A generic group algorithm $\mathcal{A}$ is a (possibly probabilistic) algorithm which takes as input a r-tuple of encoded group elements $([x_1], \ldots, [x_r])$, $x_i \in \mathbb{G}, 1 \leq i \leq r$. The algorithm may query a generic group oracle $\mathcal{O}$ to perform computation operations in some set Π and relations in some set Σ on encoded group elements.*

Normally one considers the sets $\Pi = \{\odot\}, \Sigma = \{\equiv\}$, that is the group operation and equality testing. This leads to the most basic instantiation of the GGM we can think of or at least, the most intuitive. Algorithmically, we have:

- $\mathcal{O}_{\mathbb{G}}$: The oracle for encoding group elements.
- $\mathcal{O}_{\odot}$: The oracle for performing the group operation.
- $\mathcal{O}_{\equiv}$: The oracle for testing equality of group elements.

It must be noted that these oracles are stateful with regards to elements. That is to say, if we compute the same element twice, then the results of both those computations is the same.

3 Construction

3.1 Unique Signatures Based on DDH

We now present the signature we will employ in the generic construction of our IBE scheme. Recall that the generic construction of Garg et al. [23] requires a *unique* signature scheme. When constructing such a scheme, there are two significant hurdles. Firstly, it is known that no unique signature scheme in the standard model can be tightly secure as shown by Coron [17, 18] and Kakvi and Kiltz [30, 31]. Secondly, the recent result of Döttling et al. [21] showed that no unique signature in a DDH group can be short.

Given these restrictions, it is not surprising that there is no known direct construction of DDH-based unique signatures, but only generic ones. We will use one such generic construction, specifically the tree-based signature scheme of Merkle [35]. However, we will deviate slightly from the original construction due to the fact that we cannot use standard collision-resistant hash functions, so we need to adapt our scheme to fit this. The basic building tool for this is the Discrete Logarithm-based (chameleon) hash function due to Chaum, van Heijst and Pfitzmann [13].

Hash Function based on Discrete Logarithm We now recall the hash function due to Chaum, van Heijst and Pfitzmann [13] based on the discrete logarithm problem, which will we denote as $\mathcal{CVPH}$. The hash function requires a seed $s = (g, h)$, which will be passed to the hash evaluation algorithm each time. Unlike normal hash functions that can take arbitrarily large values, $\mathcal{CVPH}$ maps two $\mathbb{Z}_p^*$ elements to one $\mathbb{Z}_p^*$ element i.e. $\mathcal{CVPH} : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$. We note that this hash function is closely related to the commitment scheme of Pedersen [38], as was used by Döttling and Garg [20], and it can be extended to larger input sizes in a similar manner. We now present the hash function in Figure 1.

Scheme $\mathcal{CVPH}$

$\underline{\mathbf{KeyGen}(1^\lambda)}$
$p \in_R \mathbb{P}[\lambda]$
$g, h \in_R \mathbb{Z}_p^*$
return $s = (p, g, h)$

$\underline{\mathbf{Hash}(s, (x_1, x_2))}$
return $y = g^{x_1} h^{x_2} \mod p$

Fig. 1. The $\mathcal{CVPH}$ Hash Function

We can see that we get collision-resistance from the discrete logarithm problem (DLog). If we can find $(x_1, x_2) \neq (y_1, y_2)$, we can find the discrete logarithm of h in base g as $\log_g h = \frac{y_1 - x_1}{x_2 - y_2}$. Recall that if DDH is hard, then DLog is also hard.

One-Time Signatures After their introduction of chameleon hash functions by Krawczyk and Rabin [33], it became clear that $\mathcal{CVPH}$ can also be viewed as a Chameleon Hash Function. We will use this fact to build a one-time signature, using the well-known method of Mohassel [36]. We will give the explicit instantiation of the one-time signature scheme, which we call the $\mathcal{CVPOTS}$ scheme. For clarity, we write the scheme mod a prime q instead of mod p so as to distinguish the two usages. We now present the one-time signature scheme in Figure 2.

Now that we have our one-time signature scheme, we can start to build it into a full-fledged signatures scheme. There are two tree-based transformations of one-time signatures to full signatures that one would consider in this situation, namely that of Merkle [35] using hash functions and that of Blazy et al. [8] using chameleon hash functions. However, neither of these approaches is directly applicable to our scenario for separate reasons. The Merkle [35] transformation relies on cryptographic collision-resistant hash functions, which is an assumption we cannot make, as it would take us out of the pure Generic Group Model. On the other hand the transformation of Blazy et al. [8] does not require this assumption, but creates signatures that are not unique. Therefore, we take ideas from both schemes and create a tree-based transformation that is tailored to our purposes.

Scheme $\mathcal{CVPOTS}$

KeyGen(1^λ)

$q \in_R \mathbb{P}[\lambda]$

$g, \alpha, \hat{m}, \hat{r} \in_R \mathbb{Z}_q^*$

$h = g^\alpha \mod q$

$z = g^{\hat{m}} h^{\hat{r}} \mod q$

return $\mathsf{vk} = (q, g, h, z), \mathsf{sk} = (\alpha, \hat{m}, \hat{r})$

Sign(sk, m)

return $\sigma = (\hat{m} - m)\alpha^{-1} + \hat{r} \mod q$

Verify(pk, m, σ)

if $(g^m h^\sigma == z \mod q)$

    return 1

else

    return 0

Fig. 2. The $\mathcal{CVPOTS}$ One-Time Signature scheme.

We first begin by outlining the general structure of both tree-based signatures. Both schemes build a binary tree of depth d and then associate each leaf with a one-time signature verification key. The leaf is then an authentication of the OTS verification key. Each parent node is the authentication of its children, leading all the way up to the root node. The root node is published as the verification key, potentially with some additional public parameters. In the scheme of Merkle [35], the authentication is the hash value, whereas in Blazy et al. [8] the authentication is a two-tier signature [6]. We will primarily follow the transformation of Merkle, with a modification at the leaves.

The primary hurdle in our setting is that the Merkle transformation relies on hash functions $\mathcal{CRHF} : \{0,1\}^* \to \{0,1\}^\ell$, which allows them to deal with values of arbitrary size. However, we have $\mathcal{CVPH} : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$, which means that we can only hash pairs of group elements. While this does fit in with the intermediate nodes, we run into the issue that our OTS verification keys consist of 3 group elements and the modulus. To get around this, we employ a hash function with a sufficiently large modulus p such that q can also be hashed, by having the bit size of p be a constant factor larger than the bit size of q (For simplicity we will use 2). We will then hash p, g and h, z to get intermediate hash values $\eta_1 = \mathcal{CVPH}.\mathsf{Hash}(s, (q, g)), \eta_2 = \mathcal{CVPH}.\mathsf{Hash}(s, (h, z))$. We then take the hash of these intermediate values to get the label of the corresponding leaf as $\mathcal{CVPH}.\mathsf{Hash}(s, (\eta_1, \eta_2))$. This is conceptually similar to just adding an additional level to our tree.

We now establish some notation for our trees. We label all nodes of our tree as $\mathtt{Node}_{\mathtt{i,j}}$ with i being the depth and j being the position, with the leftmost nodes being labelled 0 and the right most being labelled $2^i - 1$. The root node it therefore labelled as $\mathtt{Node}_{0,0}$. We also label the leaves of our tree $\mathtt{Node}_{\mathtt{d},0}, \ldots, \mathtt{Node}_{\mathtt{d},2^\mathtt{d}-1}$ as $\mathtt{Leaf}_0, \ldots, \mathtt{Leaf}_{2^\mathtt{d}-1}$ from left to right. We assume there

is en efficient algorithm that returns the path and co-path to any leaf $\texttt{Leaf}_\texttt{i}$, which we call $\texttt{TreePath}(i)$. We now present the final signature scheme in Figure 3.

Scheme $\mathcal{USIG}$

$\underline{\mathbf{KeyGen}(1^\lambda, d)}$
$s \leftarrow_\$ \mathcal{CVPH}.\mathsf{KeyGen}(1^\lambda)$

for $i \in [\![0, 2^d - 1]\!]$
  $\mathsf{vk}_i, \mathsf{sk}_i \leftarrow_\$ \mathcal{CVPOTS}.\mathsf{KeyGen}(1^{\lambda/2})$
  parse $\mathsf{vk}_i = (q_i, g_i, h_i, z_i)$
  $\eta_{i,1} = \mathcal{CVPH}.\mathsf{Hash}(s, (q_i, g_i))$
  $\eta_{i,2} = \mathcal{CVPH}.\mathsf{Hash}(s, (h_i, z_i))$
  $\mathtt{Leaf}_{\mathtt{i}} = \mathcal{CVPH}.\mathsf{Hash}(s, (\eta_{i,1}, \eta_{i,2}))$
next i

for $i \in [\![d - 1, 0]\!]$
  for $j \in [\![0, 2^i - 1]\!]$
    $\mathtt{Node}_{\mathtt{i,j}} = \mathcal{CVPH}.\mathsf{Hash}(s, (\mathtt{Node}_{\mathtt{i+1,2j-1}}, \mathtt{Node}_{\mathtt{i+1,2j}}))$
  next j
next i

$\mathsf{vk} = (s = (p, g, h), \mathtt{Node}_{0,0}), \mathsf{sk} = (\mathsf{sk}_0, \mathsf{sk}_1, \cdots, \mathsf{sk}_{2^d-1})$
return $(\mathsf{vk}, \mathsf{sk})$

$\underline{\mathbf{Sign}(\mathsf{sk}, m)}$
parse m as a integer $\in [\![0, 2^d - 1]\!]$
$(\mathtt{Node}_{0,0}, \cdots \mathtt{Leaf}_{\mathtt{m}}) \leftarrow \mathtt{TreePath}(m)$
$\hat{\sigma} = \mathcal{CVPOTS}.\mathsf{Sign}(\mathsf{sk}_m, m)$
return $\sigma = (\hat{\sigma}, \mathsf{vk}_m, \mathtt{Node}_{0,0}, \cdots \mathtt{Leaf}_{\mathtt{m}})$

$\underline{\mathbf{Verify}(\mathsf{vk}, m, \sigma)}$
parse $\mathsf{vk} = (s, \mathtt{Node}_{0,0})$
parse $\sigma = (\hat{\sigma}, \mathsf{vk}_m = (q_m, g_m, h_m, z_m), \mathtt{Node}'_{0,0}, \cdots \mathtt{Leaf}'_{\mathtt{m}})$

if $(\mathcal{CVPOTS}.\mathsf{Verify}(\mathsf{vk}_m, m, \hat{\sigma}) = 0)$
  return 0
end if

$\eta_1 = \mathcal{CVPH}.\mathsf{Hash}(s, (q_m, g_m))$
$\eta_2 = \mathcal{CVPH}.\mathsf{Hash}(s, (h_m, z_m))$
if $(\mathtt{Leaf}'_{\mathtt{m}} \neq \mathcal{CVPH}.\mathsf{Hash}(s, (\eta_1, \eta_2))$
  return 0

for $\mathtt{Node}_{\mathtt{i,j}} \in (\mathtt{Node}_{0,0}, \cdots \mathtt{Leaf}_{\mathtt{m}})$
  if $\mathtt{Node}_{\mathtt{i,j}} \neq \mathcal{CVPH}.\mathsf{Hash}(s, (\mathtt{Node}_{\mathtt{i+1,2j-1}}, \mathtt{Node}_{\mathtt{i+1,2j}})$
    return 0
next $\mathtt{Node}_{\mathtt{i,j}}$

if $(\mathtt{Node}'_{0,0} = \mathtt{Node}_{0,0})$ then
  return 1
else
  return 0

Fig. 3. The Unique Signature Scheme Based on DDH

3.2 Witness Encryption based on DDH

The next ingredient needed for our generic IBE construction is a witness encryption (WE) scheme. Since its introduction by Garg et al. [23], there have been some follow-up constructions [2, 12, 15, 46], but crucially all of these have relied on either indistinguishability obfuscation [5], or extractability obfuscation [12]. These are undesirable as the initial constructions were based on multilinear maps, which were first introduced by Boneh and Silverberg [11], which are out of the scope of the GGM. While there have been recent works on building obfuscation from other assumptions [3,4,25,29], none of these fall within the GGM.Therefore, we must look for another way to construct a witness encryption scheme.

We will use the generic construction of a Witness Encryption scheme from a Smooth Projective Hash Function (SPHF). The generic construction was proposed by Abdalla, Benhamouda and Pointcheval [1], which we now briefly recall. To encrypt a message m under an instance x generates and new set of SPHF keys and computes the hash of the instance. The hash is used as a one-time pad and is XOR'd with the message to provide a ciphertext. The ciphertext and the projective key are then sent to the receiver. Given the projective key, the receiver can recompute the hash value with the correct witness, which they can then use to decrypt the ciphertext.

Now that we have the basic idea for our Witness Encryption, we now need to define our language. For this we will work in a way similar to Katz-Wang signatures [32]. Natively our signature, are verified by checking that: $h^\sigma = g^{-m} \prod B_i^{m_i}$, instead, we are going to double those parameters, with $\hat{h}, \hat{g}, A_i$ by picking $\alpha \xleftarrow{\$} \mathbb{Z}_p$ and setting $\log_{A_i}(B_i) = \log_{\hat{h}}(h) = \log_{\hat{g}}(g) = \alpha$, intuitively now in addition to verifying the equation, $h, \hat{h}, h^\sigma, \hat{h}^\sigma$ form a DDH tuple, and the signature simply becomes a witness of this fact.

SPHF for the Verification of Signatures The language $\mathcal{L}_{\mathsf{vk}}$, we need to handle with an SPHF is the verification equation. That we know the (unique) σ such that $g^{-m} \cdot B_i(m[i]) = h^\sigma$ and $\log_{\hat{h}}(h) = \log_{\hat{g}}(g)$. The B_i is uniquely determined by the depth at which the SPHF is computed relatively to a node or a leaf.

In other words, writing $G = g^{-m} \cdot B_i^{m[i]}, \hat{G} = \hat{g}^{-m} \cdot \prod A_i^{m[i]}$, we need the witness σ such that $G, \hat{G}, h, \hat{h}$ is a DDH tuple.

- $\mathsf{HashKG}(M, \mathsf{vk})$: Outputs $\mathsf{hk} = \lambda, \mu \xleftarrow{\$} \mathbb{Z}_p$, $\mathsf{hp} = h^\lambda \hat{h}^\mu$
- $\mathsf{Hash}(\mathsf{hk}, M, \mathsf{vk})$: Outputs $G^\lambda \hat{G}^\mu$
- $\mathsf{ProjHash}(\mathsf{hp}, M, \mathsf{vk}, \sigma)$: Outputs hp^σ

This SPHF is Smooth, and it should be noted, that the person in charge of the SPHF can compute Hash without receiving data from the prover.

The Witness Encryption Scheme To achieve a Witness Encryption from an SPHF, one classically outputs a word presumably in the language, compute on Hash on it, and XOR this hash with the message to encrypt. Anyone possessing a witness the word is in the language, can then decrypt by computing the Projected Hash.

A final step is thus to combine the previous construction for every level of the tree, so $d+1$ times. For the witness encryption, we are going to need to handle the language where words are a tuple of (Identity, $\hat{h}$) , and the witness is the deterministic signature on the given identity. We use the SHPFs described above and combine them into one SPHF for our final language, which we call $\mathcal{SPHF}$.

Given an instance $x = (\mathtt{ID}, \hat{h})$, we run our combined SPHF $\mathcal{SPHF}$ to get a hashing key and a projection key. We then use the hashing key to compute the hash of the instance x, which is a single element in $\mathbb{Z}_p^*$. We use this hash as a mask and multiply it with the message, to get our masked message. Our ciphertext consists of the masked message and the projection key. Our projection key consists of $d+1$ group elements. With the masked message, and the helper generator $\hat{h}$, this gives a total ciphertext size of $d+3$ group elements to transmit.

On receipt of a ciphertext, the user can use their secret key $\mathsf{sk}_{\mathtt{ID}}$ as the witness and recompute the mask using the projection key. With this they can then unmask message, which is returned. We now give the complete witness encryption scheme $\mathcal{WE}$ below in Figure 4.

The resulted (projected) hashes are each a group element that will serve as a mask.

Scheme $\mathcal{WE}$

Encrypt$(1^\lambda, x = (\mathtt{ID}, \hat{h}), m)$

$\mathsf{prm} \leftarrow_\$ \mathcal{SPHF}.\mathsf{Setup}(1^\lambda)$

For all i:

$G_i = g^{-\mathtt{ID}} \cdot B_i(\mathtt{ID}_i)$,

$\hat{G}_i = \hat{g}^{-\mathtt{ID}} \cdot A_i(\mathtt{ID}(i))$

$\mathsf{hk}_i \leftarrow_\$ \mathcal{SPHF}.\mathsf{HashKG}(\mathcal{L}_{\mathsf{vk}})$

$\mathsf{hp}_i = \mathcal{SPHF}.\mathsf{ProjKG}(\mathsf{hk}_i)$

$H_i = \mathcal{SPHF}.\mathsf{Hash}(\mathsf{hk}_i, G_i, \hat{G}_i)$

$\hat{C} = m \cdot \prod H_i$

return $C = ((\mathsf{hp}_i)_{i\in[n+1]}, \hat{C}, \hat{h})$

Decrypt$(C, \mathtt{ID}, \sigma)$

parse $C = (\mathsf{hp}, \hat{C}, \hat{h})$

$\hat{H} = \prod \mathcal{SPHF}.\mathsf{ProjHash}(\mathsf{hp}_i, \mathcal{L}_{\mathsf{vk}}, \sigma)$

return $M = \hat{C} \cdot \hat{H}^{-1} \mod p$

Fig. 4. The Witness Encryption Scheme for $\mathcal{L}_{\mathsf{vk}}$

3.3 Modified Generic Construction

We now recall the generic construction of IBE due to Garg et al. [23]. Before we begin we must establish some notation. We define our ID space such that all identities can be expressed as bitstrings of length at most d, for some public parameter d, that is to say $\mathcal{IBE}.\mathfrak{I} \subseteq \{0,1\}^d$. Additionally, we note that we do not restrict ourselves to single bit messages, as is the case of the Witness Encryption of Garg et al. [23] or the IBE of Papakonstantinou, Rackoff, and Vahlis [37], but instead consider the more general case where messages can be longer. For simplicity we consider messages that are single group elements. We now give a high-level explanation of the scheme.

We first discuss the Setup and KeyGen algorithms together as they both rely on the same primitive. Let $\mathcal{USIG} = \{\mathcal{USIG}.\mathsf{Gen}, \mathcal{USIG}.\mathsf{Sign}, \mathcal{USIG}.\mathsf{Verify}\}$ be the secure signature scheme with unique signatures from Section 3.1 such we can sign all our identities, i.e. the depth of our tree is the same as the public parameter d. This is required as the user secret keys $\mathsf{sk}_{\mathtt{ID}}$ are simply a signature on the corresponding identity $\mathtt{ID}$. It is easy to see that the IBE public parameters pp are the signature verification key vk and the IBE master secret key msk is the signature signing key sk. Here we note that the master secret key is 2^d one-time signing keys, each of which is 3 $\mathbb{Z}_q^*$ elements. This means we have a total size of $2^d \cdot 3\lambda/ = 2^{d-1} \cdot 3\lambda$ bits. Additionally, user secret keys consist of $d+1$ elements from $\mathbb{Z}_p*$, one $\lambda/2$-bit prime q and four elements from $\mathbb{Z}_q*$. Since we know $q < p$, we pad q and all elements in $\mathbb{Z}_q*$ with leading 0's to be λ bits long for simplicity. Thus we know that our user secret keys are are $\ell = (d+6)\lambda$ bits long.

Now we discuss the $\mathsf{Encrypt}$ algorithm. We pick a random value α and compute $\hat{h} = h^\alpha$, where h is from our hash function seed. We the combine this with the ID to create an instance for the witness encryption scheme $\mathcal{WE}$ from Section 3.2. Recall our language for the witness encryption is verifying our Katz-Wang like signatures [32] with proof of membership. To encrypt, the user simply runs the WE encryption procedure and receives the ciphertext $\hat{C}$. Recall that this contains projection key, the masked message and the randomness. We combine this with our identity to form our IBE ciphertext.

Finally, we discuss the $\mathsf{Decrypt}$ algorithm. Once the the receiver has the IBE ciphertext, they can use their secret key to WE decryption algorithm to decrypt the ciphertext $\hat{C}$ and recover the message. Having given a high-level description of the IBE, we now present the full scheme.

Proof idea Our construction allows to achieve a secure IBE under the DDH assumption. As we are building on the construction of Garg et al. proof [23], the arguments of security follow in a straightforward manner. We will first provide a high level sketch, then we provide a more detail sketch proof.

Any adversary that is able to break the IBE scheme, must fall into one of two categories:

Scheme IBE-Generic

algorithm Setup(1^λ)

$(\mathsf{vk}, \mathsf{sk}) \xleftarrow{\$} \mathcal{USIG}.\mathsf{Gen}(1^\lambda, d)$
$\mathsf{pp} = \mathsf{vk}, \mathsf{msk} = \mathsf{sk}$
return $(\mathsf{pp}, \mathsf{msk})$

algorithm KeyGen$(\mathsf{msk}, \mathtt{ID})$

$\sigma \leftarrow \mathcal{USIG}.\mathsf{Sign}(\mathsf{msk}, \mathtt{ID})$
return $\mathsf{sk}_{\mathtt{ID}} = \sigma$

algorithm Encrypt$(\mathsf{pp}, \mathtt{ID}, m)$

parse $\mathsf{pp} = (p, g, h, \mathtt{Node}_{0,0})$
$\alpha \in_R \mathbb{Z}_p^*$
$\hat{h} = h^\alpha \mod p$
$\hat{C} = \mathcal{WE}.\mathsf{Encrypt}(1^\lambda, (\mathtt{ID}, \hat{h}, m)$
return $C = (\hat{C}, \mathtt{ID}, \hat{h})$

algorithm Decrypt$(\mathsf{pp}, \mathsf{sk}_{\mathtt{ID}}, C)$

parse $C = (\hat{C}, \mathtt{ID}, \hat{h})$
$x = (\mathtt{ID}, \hat{h})$
$m = \mathcal{WE}.\mathsf{Decrypt}(\hat{C}, x, \mathsf{sk}_{\mathtt{ID}})$
return m

Fig. 5. The Generic Construction of an IBE scheme [23]

1. *The adversary is able to generate secret keys themselves.* Since the adversary never sees the secret key for the target identity ID*, this is equivalent to forging a signature, which in turn would break the DLog assumption.
2. *The adversary is able to distinguish the encryption of their message from a random encryption.* This is directly equivalent to breaking the underlying WE built from the SPHF, which in turn would break the DDH assumption.

We now give a brief sketch as to how we achieve this. In the first game hop, we would move from generating user secret keys to querying the signature security game for all the ID requests. This way, we are able to provide a user secret key oracle without the master secret key. Here the challenge ciphertext would be generated normally. From the view of the adversary this is identical to the normal game.

The next game hop would be to switch the generator $\hat{h}$ to random value so that $(G, \hat{G}, h, \hat{h})$ is no longer a valid DDH tuple. The difference to the previous game is negligable, under the DDH assumption, as a significant advantage loss compared to the previous game would imply a distinguisher for DDH challenges.

Additionally in this game, there is are valid signature that would be a valid witness for any statements, hence the adversary can do no better than random guessing. Thus the advantage of the adversary in the game is effectively 0.

Detailed Proof

Theorem 5. *Our construction allows to achieve a secure IBE under the DDH assumption*

Proof. In this proof, we are going to build a simulator $\mathcal{B}$ using an adversary $\mathcal{A}$ against our IBE to either break the unforgeability or our signature scheme, or the pseudo-randomness of the underlying SPHF. In both cases, this would lead to breaking a DDH challenge.

We start from a game $\mathcal{G}_0$. This is the real game, everything is generated honestly. The adversary is allowed to query user secret keys, and wins if he can decrypt a challenge ciphertext on a chosen identity if and only if he never queries a user secret key for this id. $\mathsf{Adv}^{\mathbb{G}_0} = \mathsf{Adv}^{\mathsf{real}}$

In game $\mathcal{G}_1$, we now use the signature Osign oracle to answer the $\mathsf{UserKey}$ Queries. As, the challenge identity is not allowed to be queries, this game is indistinguishable of the previous one (under appropriate simulation in the signature security proof). $|\mathsf{Adv}^{\mathbb{G}_1} - \mathsf{Adv}^{\mathbb{G}_0}| \leq \mathsf{Adv}_{\mathsf{uf}}$

In game $\mathcal{G}_2$, we forget the signature secret key we were no longer using. As this is just some internal memory state of the simulator this is strictly equivalent to the previous game. $\mathsf{Adv}^{\mathbb{G}_2} = \mathsf{Adv}^{\mathbb{G}_1}$

In game $\mathcal{G}_3$, we now alter the challenge ciphertext. Instead of using a word in the language, we switch it to outside the language (as in the Katz-Wang signatures), hence the adversary has to lose the game (except with neglibile probability). We use the secret Hash key hk from the sphf to produce the valid output. Under DDH, this game is indistinguishable from the previous one. $|\mathsf{Adv}^{\mathbb{G}_3} - \mathsf{Adv}^{\mathbb{G}_2}| \leq \mathsf{Adv}^{\mathsf{DDH}}$

At this stage, there is no valid user secret key possible for the description, hence, the security relies purely on the smoothness of the SPHF. And so $\mathsf{Adv}^{\mathbb{G}_3} \leq \varepsilon$. □

Which leads to the conclusion $\mathsf{Adv}^{\mathsf{real}} \leq \mathsf{Adv}^{\mathsf{uf}} + \mathsf{Adv}^{\mathsf{DDH}} + \varepsilon = \mathcal{O}(\mathsf{Adv}^{\mathsf{DDH}})$

4 Discussion

Now that we have shown that we can construct an IBE in the GGM, albeit a somewhat inefficient one, we now consider how this affects the PVR impossibility result. It would seem at first glance that the PVR result is incorrect as we have proven a result directly contrary to theirs. While at a high level this does seem to be true, as there are several subtle details of the PVR impossibility which we need to examine more closely to get a more complete answer. We will give a high level overview of the PVR impossibility result and then we will show where problems may arise and how we provide an alternative formulation of the result. We believe that the result is not a general impossibility result, but more likely rules out *tightly secure and efficient* constructions. This is in line with the recent results of Döttling et al. [21] who show that no unique signature can be "short" using similar techniques. We now recap the PVR impossibility result.

4.1 The PVR Impossibility Result [37]

We will now give a high-level overview of the PVR impossibility result and discuss how it relates to our construction. For a more detailed description, we refer the reader to the original paper [37]. We begin by introducing the notation needed for the PVR impossibility result. In the impossibility result, IBE schemes are parametrised by the prime modulus p and 3 additional values $m, n \in \mathbb{Z}, \varepsilon \in (0.5, 1]$. The value n is the bit-size of the master secret key msk and the bit size of the randomness used in encryption. The value m is the maximum number of GGM oracle queries by any of the IBE algorithms and is also the upper bound on the number of group elements output by any IBE algorithm. Finally, ε is the correctness error; that is to say, a valid ciphertext is correctly decrypted with probability $\geq \varepsilon$.

The first step in the impossibility result is to transform an IBE scheme to a so-called Restricted IBE (RIBE), which has no group elements in the secret key. To this end, Papakonstantinou, Rackoff, and Vahlis introduce a transformation that turns any (p, m, n, ε)-IBE scheme into a $(p, \mathrm{poly}(m), \mathrm{poly}(n), \varepsilon - \frac{1}{\mathrm{poly}(n)})$-RIBE [37]. The RIBE key generation algorithm first runs the IBE key generation algorithm to get $\mathsf{sk}_{\mathtt{ID}}$. The algorithm then generates a large number of ciphertexts and uses $\mathsf{sk}_{\mathtt{ID}}$ to decrypt them. By observing the queries and responses made during these decryption procedures the algorithm can rewrite all the secret key elements as sums of the public key elements and the ciphertext elements. Using this, the secret key elements can all be rewritten as these sums of public elements. The decryption procedure is modified to accept these new keys. It must be noted that any RIBE that results from this transformation, as some of the secret information may be lost in the transformation. However, the resulting RIBE is a secure IBE scheme.

The next step is to show an attacker against a (p, m, n, ε)-RIBE. For this attack, we have an additional parameter $c > 0$. The RIBE attacker receives $k_1 \in_R [\![n^{2c}, 2n^{2c}]\!]$ user secret keys $\mathsf{sk}_{\mathtt{ID}_1}, \ldots, \mathsf{sk}_{\mathtt{ID}_{k_1}}$ for identities $\mathtt{ID}_1, \ldots, \mathtt{ID}_{k_1}$ of its choice and a challenge ciphertext C^* under the identity $\mathtt{ID}^*$ of its choice. It uses these secret keys to construct a partial master secret key msk'. The adversary now generates a large number of ciphertexts for some other identities $\mathtt{ID}_{k_1+1}, \ldots, \mathtt{ID}_{k_1+k_2}$, with $k_2 \in_R \in_R [\![n^{2c}, 2n^{2c}]\!]$, which it uses to learn "frequently accessed elements". With all this information the adversary builds a simulated secret key $\mathsf{sk}_{\mathtt{ID}*}$ using the information it has collected. Once it has a simulated secret key $\mathsf{sk}_{\mathtt{ID}}*$, it attempts to decrypt the challenge ciphertext. The adversary succeeds with a probability equal to the correctness error of the RIBE. We now recall the theorem statement of the attacker.

Theorem 6 (Thm. 1 in [37]). *Let $\mathcal{RIBE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a restricted (p, m, n, ε)-IBE, then for every $c > 0$ and sufficiently large n, there exists and adversary which breaks the security of $\mathcal{RIBE}$ with poly(m, n) queries and advantage $\varepsilon - \frac{1}{2} - \frac{1}{n^c}$.*

4.2 Shortcomings of the the PVR Impossibility Result

We now look at how the PVR impossibility result works with our scheme and where it fails. First, we establish what our parameters are. We see that $n = |\mathsf{msk}| = 2^d \cdot 3 \cdot \lambda/2 = 2^{d-1} \cdot 3\lambda$, as we have 2^d one time signing keys, each with 3 elements of size $\lambda/2$. Next, we have that $m = |C| = d+3$. Finally it is clear to see we have $\varepsilon = 1$. Even before we apply the RIBE transformation, a problem starts to become evident. The attacker requires k_1 secret keys, where $k_1 \in [\![n^{2c}, 2n^{2c}]\!]$, which if we plug in our value of n and use the minimal value $c = 1$, this gives us $k_1 \in [\![(2^{d-1} \cdot 3\lambda)^2, 2(2^{d-1} \cdot 3\lambda)^2]\!] = [\![2^{2d-2} \cdot 9\lambda^2, 2^{2d-1} \cdot 18\lambda^2]\!]$, which is larger than the total number of valid user secret keys 2^d. While at first glance it seems possible that we could reduce this by using smaller moduli for our signatures, but we would have to make them so small as to be impractical. Furthermore, we note that these figures are for our base IBE and not for the reduced IBE that results from the transformation, where the parameters are polynomial in the original parameters.

Therefore, it is clear that the PVR does not apply to our case. However, based on this it is not possible to completely discount the result, as our scheme does not expose any flaws in the core methodology of the attack. It seems most likely that the result only applies to "compact" IBEs, for some appropriate definition of compact. This seems the most likely result, as it fits in perfectly with the recent impossibility by Döttling et al. [21], which states that there are no "short" unique signatures in the GGM. The impossibility of "compact" IBEs combined with the generic construction of Garg et al. [23], immediately implies the impossibility of "short" unique signatures, as shown by Döttling et al. [21].

This idea is further reinforced by the number of oracle queries that are required be the adversary. The attacker against a (p, n, m, ε)-RIBE requires $\mathrm{poly}(n, m)$ queries to the oracles. As n and m, this number also grows, especially considering both n and m are polynomially larger than the parameters from the original IBE. As the number of queries approaches $\sqrt{p}$, the RIBE attack starts to be less efficient than the generic attacks on DDH/DLog, such as that of Pollard [39]. Therefore, for the attack to be a valid attack on the IBE system, the parameters n, m must be as small as possible, that is to say the IBE must be "compact". Therefore, it seems that the PVR impossibility result only holds for "compact" IBEs.

Another aspect that is overlooked by the PVR impossibility result is the tightness of the security reduction. If the IBE is tightly secure, then any attack on the IBE would result in an attack on the underlying assumption, in our case DDH and/or DLog. However, our scheme is not tight and our loss is in the number in secret key queries, which are signing queries to our unique signature. Therefore even if the attack were successful against our IBE scheme, it would not necessarily lead to an attack on DDH/DLog. This is further compounded by the fact that the attack is not always successful. While it might be tempting to assume that this rules out any tight IBE in the GGM, it is not clear that this is the case. The loss in our scheme is unavoidable, as any unique signature cannot

be tight, but that does not mean that there does not exist an IBE scheme where there is no loss.

5 Conclusions

We have constructed the first IBE scheme in a DDH only group, by using the generic construction of Garg et al. [23]. To do this, we showed the first unique signature scheme in DDH only groups and combined that with known results on SPHFs to obtain our IBE. The resulting IBE is not very efficient, but serves as a counter-example to the impossibility result of Papakonstantinou, Rackoff, and Vahlis [37]. We showed that while our construction contradicts the impossibility result, it does not fully negate it and it seems that result only rules out all IBEs with small parameters. This is reinforced by the impossibility result of short signatures by Döttling et al. [21]. Thus, while the PVR impossibility result does not rule out *all possible* IBEs in DDH only groups, it does rule out all *practical* IBEs. Therefore, while the result is mildly overstated, it is still correct for all practical purposes.

References

1. Abdalla, M., Benhamouda, F., Pointcheval, D.: Disjunctions for hash proof systems: New constructions and applications. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 69–100. Springer, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6˙3
2. Abusalah, H., Fuchsbauer, G., Pietrzak, K.: Offline witness encryption. In: Manulis, M., Sadeghi, A.R., Schneider, S. (eds.) ACNS 16. LNCS, vol. 9696, pp. 285–303. Springer, Heidelberg (Jun 2016). https://doi.org/10.1007/978-3-319-39555-5˙16
3. Agrawal, S.: Indistinguishability obfuscation without multilinear maps: New methods for bootstrapping and instantiation. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 191–225. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2˙7
4. Ananth, P., Jain, A., Lin, H., Matt, C., Sahai, A.: Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 284–332. Springer, Heidelberg (Aug 2019). https://doi.org/10.1007/978-3-030-26954-8˙10
5. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S.P., Yang, K.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8˙1
6. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 201–216. Springer, Heidelberg (Apr 2007). https://doi.org/10.1007/978-3-540-71677-8˙14
7. Benhamouda, F., Blazy, O., Ducas, L., Quach, W.: Hash proof systems over lattices revisited. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 644–674. Springer, Heidelberg (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5˙22

8. Blazy, O., Kakvi, S.A., Kiltz, E., Pan, J.: Tightly-secure signatures from chameleon hash functions. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 256–279. Springer, Heidelberg (Mar / Apr 2015). https://doi.org/10.1007/978-3-662-46447-2˙12
9. Blazy, O., Kiltz, E., Pan, J.: (Hierarchical) identity-based encryption from affine message authentication. In: Garay, J.A., Gennaro, R. (eds.) CRYPTO 2014, Part I. LNCS, vol. 8616, pp. 408–425. Springer, Heidelberg (Aug 2014). https://doi.org/10.1007/978-3-662-44371-2˙23
10. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001). https://doi.org/10.1007/3-540-44647-8˙13
11. Boneh, D., Silverberg, A.: Applications of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080 (2002), https://eprint.iacr.org/2002/080
12. Boyle, E., Chung, K.M., Pass, R.: On extractability obfuscation. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 52–73. Springer, Heidelberg (Feb 2014). https://doi.org/10.1007/978-3-642-54242-8˙3
13. Chaum, D., van Heijst, E., Pfitzmann, B.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1˙38
14. Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (Aug 2013). https://doi.org/10.1007/978-3-642-40084-1˙25
15. Chvojka, P., Jager, T., Kakvi, S.A.: Offline witness encryption with semi-adaptive security. In: Conti, M., Zhou, J., Casalicchio, E., Spognardi, A. (eds.) ACNS 20, Part I. LNCS, vol. 12146, pp. 231–250. Springer, Heidelberg (Oct 2020). https://doi.org/10.1007/978-3-030-57808-4˙12
16. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) 8th IMA International Conference on Cryptography and Coding. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (Dec 2001)
17. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. Cryptology ePrint Archive, Report 2001/062 (2001), https://eprint.iacr.org/2001/062
18. Coron, J.S.: Optimal security proofs for PSS and other signature schemes. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 272–287. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7˙18
19. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 45–64. Springer, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7˙4
20. Döttling, N., Garg, S.: Identity-based encryption from the Diffie-Hellman assumption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 537–569. Springer, Heidelberg (Aug 2017). https://doi.org/10.1007/978-3-319-63688-7˙18
21. Döttling, N., Hartmann, D., Hofheinz, D., Kiltz, E., Schäge, S., Ursu, B.: On the impossibility of short algebraic signatures. Cryptology ePrint Archive, Report 2021/738 (2021), https://ia.cr/2021/738
22. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory **31**, 469–472 (1985)

23. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Boneh, D., Roughgarden, T., Feigenbaum, J. (eds.) 45th ACM STOC. pp. 467–476. ACM Press (Jun 2013). https://doi.org/10.1145/2488608.2488667
24. Gennaro, R., Lindell, Y.: A framework for password-based authenticated key exchange. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 524–543. Springer, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9˙33, https://eprint.iacr.org/2003/032.ps.gz
25. Gentry, C., Jutla, C.S., Kane, D.: Obfuscation using tensor products. Cryptology ePrint Archive, Report 2018/756 (2018), https://eprint.iacr.org/2018/756
26. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407
27. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: 21st ACM STOC. pp. 25–32. ACM Press (May 1989). https://doi.org/10.1145/73007.73010
28. Jager, T., Schwenk, J.: On the analysis of cryptographic assumptions in the generic ring model. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 399–416. Springer, Heidelberg (Dec 2009). https://doi.org/10.1007/978-3-642-10366-7˙24
29. Jain, A., Lin, H., Matt, C., Sahai, A.: How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 251–281. Springer, Heidelberg (May 2019). https://doi.org/10.1007/978-3-030-17653-2˙9
30. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 537–553. Springer, Heidelberg (Apr 2012). https://doi.org/10.1007/978-3-642-29011-4˙32
31. Kakvi, S.A., Kiltz, E.: Optimal security proofs for full domain hash, revisited. Journal of Cryptology **31**(1), 276–306 (Jan 2018). https://doi.org/10.1007/s00145-017-9257-9
32. Katz, J., Wang, N.: Efficiency improvements for signature schemes with tight security reductions. In: Jajodia, S., Atluri, V., Jaeger, T. (eds.) ACM CCS 2003. pp. 155–164. ACM Press (Oct 2003). https://doi.org/10.1145/948109.948132
33. Krawczyk, H., Rabin, T.: Chameleon signatures. In: NDSS 2000. The Internet Society (Feb 2000)
34. Maurer, U.M.: Abstract models of computation in cryptography (invited paper). In: Smart, N.P. (ed.) 10th IMA International Conference on Cryptography and Coding. LNCS, vol. 3796, pp. 1–12. Springer, Heidelberg (Dec 2005)
35. Merkle, R.C.: A certified digital signature. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 218–238. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0˙21
36. Mohassel, P.: One-time signatures and chameleon hash functions. In: Biryukov, A., Gong, G., Stinson, D.R. (eds.) SAC 2010. LNCS, vol. 6544, pp. 302–319. Springer, Heidelberg (Aug 2011). https://doi.org/10.1007/978-3-642-19574-7˙21
37. Papakonstantinou, P.A., Rackoff, C.W., Vahlis, Y.: How powerful are the DDH hard groups? Cryptology ePrint Archive, Report 2012/653 (2012), https://eprint.iacr.org/2012/653
38. Pedersen, T.P.: Non-interactive and information-theoretic secure verifiable secret sharing. In: Feigenbaum, J. (ed.) CRYPTO'91. LNCS, vol. 576, pp. 129–140. Springer, Heidelberg (Aug 1992). https://doi.org/10.1007/3-540-46766-1˙9
39. Pollard, J.M.: A monte carlo method for factorization. BIT Numerical Mathematics **15**(3), 331–334 (1975)

40. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairing. In: SCIS 2000. Okinawa, Japan (Jan 2000)
41. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Brassard, G. (ed.) CRYPTO'89. LNCS, vol. 435, pp. 239–252. Springer, Heidelberg (Aug 1990). https://doi.org/10.1007/0-387-34805-0˙22
42. Secretary), C.F.K.A., (Director), C.R.: Fips pub 186-4 federal information processing standards publication digital signature standard (dss) (2013)
43. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (Aug 1984)
44. Shoup, V.: Lower bounds for discrete logarithms and related problems. In: Fumy, W. (ed.) EUROCRYPT'97. LNCS, vol. 1233, pp. 256–266. Springer, Heidelberg (May 1997). https://doi.org/10.1007/3-540-69053-0˙18
45. Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (May 2005). https://doi.org/10.1007/11426639˙7
46. Zhandry, M.: How to avoid obfuscation using witness PRFs. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A, Part II. LNCS, vol. 9563, pp. 421–448. Springer, Heidelberg (Jan 2016). https://doi.org/10.1007/978-3-662-49099-0˙16