Towards Witness Encryption Without Multilinear Maps

Extractable Witness Encryption for Multi-Subset Sum Instances with no Small Solution to the Homogeneous Problem

Gwangbae Choi[1] and Serge Vaudenay[2]

[1] Fasoo
Seoul, Korea
gwangbae.choi@hotmail.com
[2] Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
serge.vaudenay@epfl.ch

Abstract. Current proposals of extractable witness encryption are based on multilinear maps. In this paper, we propose a new construction without.
We propose the notion of hidden group with hashing and make an extractable witness encryption from it. We show that the construction is secure in a generic model. We propose a concrete construction based on RSA-related problems. Namely, we use an extension of the knowledge-of-exponent assumption and the order problem. Our construction allows to encrypt for an instance of the subset sum problem (actually, a multi-dimensional variant of it) for which short solutions to the homogeneous equation are hard to find. Alas, we do not propose any reduction from a known NP-complete problem.

Keywords: Witness key encapsulation mechanism, Subset sum problem

1 Introduction

Witness encryption was first proposed by Garg et al. [10]. The idea is that a secret is encrypted together with an instance x of an NP language. The resulted ciphertext can be decrypted by using a witness ω for the instance, which is verified by a relation $R(x, \omega)$.

Witness encryption based on an NP-complete language is a powerful primitive as it implies a witness encryption based on any NP language. Anyone can encrypt a message for anyone who could solve a given equation $R(x, .)$. This is very nice to encrypt a bounty. It can also be used to send secrets to the future [14].

There are several kinds of witness encryption schemes. Regular schemes offer IND-CPA security when the encryption key x does *not* belong to the language. However, in that case, decryption is not possible either. Extractable schemes are such that for any efficient adversary, there must exist an efficient extractor such that either it is hard for the adversary to decrypt, or it is easy for the extractor having the same inputs to produce a witness. Like obfuscation, existing constructions of extractable witness encryption are based on multilinear maps which are currently heavy algorithms. To mitigate their complexity, offline schemes allow efficient encryption but have an additional setup algorithm which does the heavy part of the scheme.

Cramer and Shoup proposed the notion of Hash-proof systems which is also based on NP languages [5]. Those systems use a special hash function which has a public key and a secret key. We can hash an instance x either with its witness ω together with the public key, or with the secret key alone. Somehow, the secret key is a wildcard for a missing witness. Hash-proof systems are used to build *CCA-secure KEM* [6]. We encapsulate by picking a random (x, ω) pair in the relation R and hashing x to obtain a key $K = h_{\mathsf{pk},\omega}(x)$ and it encapsulates into $\mathsf{ct} = x$. We decapsulate using the secret key: $K = h_{\mathsf{sk}}(\mathsf{ct})$. In witness encryption, the construction is upside down: we encapsulate with x by generating a fresh key pair $(\mathsf{sk}, \mathsf{pk})$ for the hash-proof system and we hash using the secret key: $K = h_{\mathsf{sk}}(x)$ and $\mathsf{ct} = \mathsf{pk}$. We decapsulate by hashing with a witness and the public key: $K = h_{\mathsf{ct},\omega}(x)$ One problem is to build a hash-proof system with extractable security for an NP-complete problem.

The notion of security with extractor of the witness encryption is non-falsifiable [15]. There exist other non-falsifiable notions which use extractors. For instance, the knowledge-of-exponent assumption (KEA) was proposed by Damgård in 1991 [7]. It says that for any efficient adversary, there must exist an efficient extractor such that given (g, g^y) in a given group, it is hard, either for the adversary to construct a pair of form (g^x, g^{xy}), or for the extractor having the same input not to produce x. KEA can be proven in the *generic group model* [1, 8].

Witness encryption can be achieved using obfuscation: the ciphertext is an obfuscated program which takes as input ω and releases the plaintext if $R(x, \omega)$ holds. As shown by Chvojka et al. [3], this can also be turned into an offline witness encryption scheme. An alternate approach from Faonio et al. [9] relies on *predictable arguments of knowledge*. It was used by Barta et al. [2] to construct a solution based on the approximation problem for the minimal distance of a linear code.

Our contribution. In this paper, we construct an efficient witness encryption scheme[3] WKEM for a variant of the subset sum problem. Concretely, an instance is a tuple $x = (x_1, \ldots, x_t, s)$ of vectors x_i and s, a witness is a vector $\omega = (a_1, \ldots, a_t)$ of non-negative *small* integers a_i, and $R(x, \omega)$ is equivalent to the vectorial equation $a_1 x_1 + \cdots + a_t x_t = s$. In the regular subset sum problem, all a_i must be boolean and the vectors x_i and s are actually scalars (i.e. the dimension is $d = 1$). Here, we require the a_i to be polynomially bounded and vectors have some dimension d. We also require the homogeneous equation $a'_1 x_1 + \cdots + a'_t x_t = 0$ to have no small integer solution a'_i (positive or negative), which is a severe limitation of our construction. So, we require x to belong to a language $L_1 \cap L_2$ with $L_1 \in$ NP (the Multi-SS problem) and $L_2 \in$ coNP (the co-HLE problem). Alternately, we require such homogeneous relation to be hard to find.

Our encryption scheme is based on the following idea which we explain for $d = 1$ as follows: encryption generates a (n, ℓ, g, k) tuple such that g has multiplicative order ℓ modulo n and k is invertible modulo ℓ. The values k and ℓ are not revealed. Then, the ciphertext consists of $(n, g, y_1, \ldots, y_t)$ with $y_i = k^{x_i} \bmod \ell$ and the encapsulated key is $h = g^{k^s} \bmod n$. It is believed that given a set of $(x_i, k^{x_i} \bmod \ell)$ pairs with large x_i, it is hard to recover a multiple of ℓ, even when given (n, g, h), and unless a linear relation with small coefficients is known about the x_i. The decryption rebuilds $h = g^{y_1^{a_1} \cdots y_t^{a_t}} \bmod n$ from the ciphertext and the witness. The key idea in the security is that the operations need to be done in the hidden group of residues modulo ℓ. The product $y_1^{a_1} \cdots y_t^{a_t}$ can only be done over the integers since ℓ is unknown, and is feasible because the a_i's are small. However, the basis-g exponential reduces it modulo ℓ in a hidden manner.

Interestingly, computing the products $y_1^{a_1} \cdots y_t^{a_t}$ from reduced y_i values resembles to the notion of *graded encoding*, which is the basis of currently existing multilinear maps. In our construction, a 1-level encoding of x is $k^x \bmod \ell$. Hence, each y_i is a 1-level encoding of x_i and $y_1^{a_1} \cdots y_t^{a_t}$ is an encoding of $a_1 x_1 + \cdots + a_t x_t$ of level $a_1 + \cdots + a_t$. The level of encoding is somehow proportional to the size of the integer.

Our construction is based on a homomorphism mapping x_i to y_i from $\mathbb{Z}^d$ to the hidden group $\mathbb{Z}_\ell^*$. This hidden group is included in a larger structure $\mathbb{Z}$. We can do multiplications in $\mathbb{Z}$ which are compatible with the hidden group. However, we later need to reduce elements in a compatible and hidden manner. We call this reduction operation *hashing*. In our

[3] Actually, we construct a KEM.

construction, it is done by the $y \mapsto g^y \bmod n$ function. We formalize the notion of hidden group with hashing (HiGH).

To be able to prove security, we need an assumption which generalizes the knowledge-of-exponent assumption: we need to say that computing h implies being able to write it as the exponential of some (multiplicative) linear combination of the y_i's with known exponents. To do so, we must make the group sparse over the integers (so that we cannot find element by chance). For that, we duplicate the basis-k exponential like in the Cramer-Shoup techniques [4]. Then, we formulate two computational assumptions. The first one, which we call the *kernel assumption* says that it is hard to find a non-zero vector x mapping to 1 by the homomorphism, with only public information (i.e., the ciphertext). We show that it is equivalent to the order assumption for the RSA modulus ℓ: given a random $k \in \mathbf{Z}_\ell^*$, it is hard to find a multiple of the order of k. The second one, which is non-standard, is similar to the knowledge of exponent assumption, and so is non-falsifiable. The game will be defined as the HiGH-KE game in the paper. A simplification of this game (in dimension $d = 1$) for our favorite instance looks like what follows:

Input: x:

1: parse $x = (x_1, \ldots, x_t, \mathsf{aux})$
2: pick RSA moduli ℓ and $n = pq$ such that ℓ divides $p - 1$ and $q - 1$
3: pick g of order ℓ in $\mathbf{Z}_n^*$
4: pick $k \in \mathbf{Z}_\ell^*$ and $\theta \in \mathbf{Z}_{\varphi(\ell)}^*$
5: $y_i \leftarrow (k^{x_i}, k^{\theta x_i}) \bmod \ell$, $i = 1, \ldots, t$
6: $\mathcal{A}'(x_1, \ldots, x_t, y_1, \ldots, y_t, n, g, \mathsf{aux}) \to h$
7: **if** there is no ξ such that $h = (g^{k^\xi}, g^{k^{\theta\xi}}) \bmod n$ **then** abort
8: set ρ to the random coins used by $\mathcal{A}'$
9: $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, n, g, \mathsf{aux}, \rho) \to (1^{a_1}, \ldots, 1^{a_t})$
10: **if** $h = (g^{k^{a_1 x_1 + \cdots + a_t x_t}}, g^{k^{\theta(a_1 x_1 + \cdots + a_t x_t)}}) \bmod n$ **then return** 0
11: **return** 1

Essentially, we want that for every adversary $\mathcal{A}'$, there exists an extractor $\mathcal{E}'$ such that if $\mathcal{A}'$ succeeds to forge the exponential of a pair of form $(k^\xi, k^{\theta\xi})$, then the extractor finds ξ as a linear combination $\xi = a_1 x_1 + \cdots + a_t x_t \bmod \ell$ with small non-negative integers a_i. In other words, the only way to forge such a pair is to pick some small a_i and to compute $y_1^{a_1} \cdots y_t^{a_t}$ over the integers (because ℓ is not hidden).

We prove the security in a generic HiGH model. We also propose an RSA-based HiGH for which we prove security (under our non-standard

but realistic assumptions but without the generic model) for instances x which have no $a_1x_1 + \cdots + a_tx_t = 0$ relation with small a_i.

Structure of this paper. We start with preliminaries in Section 2. We define NP languages, the subset sum problem SS, the multidimensional subset sum problem Multi-SS, and the homogeneous linear equation problem HLE. In Section 3, we define WKEM, a witness key encapsulation mechanism. We define the extractable security notions extractable-OW and IND-extractable. We show that IND-extractable is a stronger security notion than extractable-OW and we show how to construct an IND-extractable WKEM from an extractable-OW WKEM using a random oracle. In Section 4, we define our notion of Hidden Group With Hashing (HiGH). We prove basic properties and define two security notions for HiGH: the knowledge exponent assumption (HiGH-KE) and the kernel assumption (HiGH-Ker). In Section 5, we propose a generic construction of an extractable WKEM from a HiGH satisfying both properties. In Section 6, we propose a construction of a HiGH based on RSA. We finally conclude. Due to lack of space, some proofs are provided in the full version of this paper. Our full version also includes a definition for a generic HiGH model and prove security in this model.

2 Preliminaries

We denote the indicator function by $\mathbb{1}_r$. We consider "words" as bitstrings (i.e. we use a binary alphabet) and $|x|$ denotes the bit length of x. 1^a is the bitstring of length a with all bits set to 1. $\#S$ denotes the cardinality of the set S. $\mathsf{negl}(\lambda)$ denotes any function f such that for all $c > 0$, for any sufficiently large λ, we have $|f(\lambda)| < \frac{1}{\lambda^c}$. Similarly, $\mathsf{Poly}(\lambda)$ denotes any function f such that there exists $c > 0$ such that for any sufficiently large λ, we have $|f(\lambda)| < \lambda^c$. For simplicity, all advantages are considered as a function of λ which is omitted.

Definition 1 (NP language). *Let L be a language. The language L is in the class NP if there exists a predicate R and a polynomial P such that L is the set of all words x for which there exists a witness ω satisfying $R(x, \omega)$ and $|\omega| \leq P(|x|)$, and if we can compute R in time polynomially bounded in terms of the size of x.*

It is important to stress that in what follows, the predicate is actually more important than the language itself.

Our construction will be based on a variation Multi-SS of the subset sum problem SS. We first define the subset sum problem. Intuitively, the

subset sum problem is a problem of finding a subset of a given set of integers whose sum is equal to a target value. The Subset Sum (SS) NP language is defined by:

> **Instance**: a tuple $x = (x_1, \ldots, x_t, s)$ of non-negative integers.
> **Witness**: a tuple $\omega = (a_1, \ldots, a_t)$ of bits $a_i \in \{0, 1\}$, $i = 1, \ldots, t$.
> **Predicate** $R(x, \omega)$: $a_1 x_1 + \cdots + a_t x_t = s$.

It is well-known that SS is NP-complete [13]. We extend SS to the Multi-SS predicate R in dimension d by:

> **Instance**: a tuple $x = (x_1, \ldots, x_t, s)$ of *vectors* of non-negative integers in $\mathbb{Z}^d$.
> **Witness**: a tuple $\omega = (1^{a_1}, \ldots, 1^{a_t})$ with non-negative integers a_i, $i = 1, \ldots, t$.
> **Predicate** $R(x, \omega)$: $a_1 x_1 + \cdots + a_t x_t = s$.

In Multi-SS, we write $\omega = (1^{a_1}, \ldots, 1^{a_t})$ to stress that the a_i must be polynomially bounded in terms of $|x|$. It is easy to show that for $d \geq 1$, over the space $\mathbb{Z}^d$, the problem is NP-complete. We give here a similar reduction as the one by Groth et al. [12]:

1. Start from SAT which is NP-complete.
2. Reduce to a system of Boolean equations, all of form u NOR $v = w$.
3. Reduce to a system of linear equations over $\mathbb{N}$ with positive integral coefficients.
   - Each Boolean literal z is mapped to a pair of unknowns (z_+, z_-) coming with a linear equation $z_+ + z_- = 1$.
   - Each u NOR $v = w$ equation is mapped to a pair of unknowns (g_+, g_-) coming with a linear equation $g_+ + g_- = 1$.
   - Each u NOR $v = w$ equation is mapped to a linear equation $u_+ + v_+ + g_+ + 2w_- = 2$.
4. Reduce to Multi-SS by writing the system of equations as $X \times a = s$ where X is a $d \times t$ matrix of coefficients in $\{0, 1, 2\}$, a is a vector of t unknowns, and s is a vector of d coefficients in $\{1, 2\}$.

Hence, Multi-SS seeks vectors a_i of non-negatives as opposed to Booleans a_i for SS. Unfortunately, this reduction introduces short solutions to the homogeneous problem like $z_+ = g_+ = +1$ and $z_- = g_- = -1$ for all z and g. The problem is that such solution will make our construction insecure. Namely, we consider the *Homogeneous Linear Equation* problem (HLE):

Instance: a tuple $x = (x_1, \ldots, x_t)$ of *vectors* in $\mathbb{Z}^d$.
Witness: a tuple $\omega = (1^{a_1}, \ldots, 1^{a_t}, b_1, \ldots, b_t)$ with non-negative integers a_i and bits $b_i \in \{0, 1\}$, $i = 1, \ldots, t$, with $(a_1, \ldots, a_t) \neq (0, \ldots, 0)$.
Predicate $R(x, \omega)$: $(-1)^{b_1} a_1 x_1 + \cdots + (-1)^{b_t} a_t x_t = 0$.

3 Primitives of Witness Key Encapsulation Mechanism

We adapt the primitives of witness encryption from Garg et al. [10] so that we have a key encapsulation mechanism instead of a cryptosystem.

Definition 2 (Witness key encapsulation mechanism (WKEM)). *Let R be an NP predicate. A witness key encapsulation mechanism for R consists of the following two algorithms and a domain $\mathcal{K}_\lambda$ defined by a security parameter λ:*

- $\mathsf{Enc}(1^\lambda, x) \to K, \mathsf{ct}$*: A probabilistic polynomial-time algorithm which takes a security parameter λ and a word x as inputs, and outputs a plaintext $K \in \mathcal{K}_\lambda$ and a ciphertext* ct.
- $\mathsf{Dec}(\omega, \mathsf{ct}) \to K/\perp$*: A deterministic polynomial-time algorithm which takes a witness ω and a ciphertext* ct *as inputs, and outputs a plaintext K or $\perp$ which indicates the decryption failure.*

Then, the following property is satisfied:

- ***Correctness****: For any security parameter λ, for any word x and witness ω such that $R(x, \omega)$ is true, we have*

$$\Pr_{\gamma}\left[\mathsf{Dec}(\omega, \mathsf{ct}) = K | (K, \mathsf{ct}) \leftarrow \mathsf{Enc}(1^\lambda, x; \gamma)\right] = 1.$$

Based on the security notions of extractable witness encryption [11] and KEM [6], we define extractable indistinguishability as follows.

Definition 3 (Extractable indistinguishability). *Let $(\mathcal{K}_\lambda, \mathsf{Enc}, \mathsf{Dec})$ be a* WKEM *for R. Given an adversary $\mathcal{A}$, we define the following game with $b \in \{0, 1\}$:*

Game $\mathsf{IND\text{-}EWE}^b_{\mathcal{A}}(1^\lambda, x)$*:*

1: $\mathsf{Enc}(1^\lambda, x) \to K_1, \mathsf{ct}$
2: *pick a random* $K_0 \in \mathcal{K}_\lambda$
3: $\mathcal{A}(x, K_b, \mathsf{ct}) \to r$
4: ***return*** r

We define the advantage of $\mathcal{A}$ by

$$\mathsf{Adv}^{\mathsf{IND\text{-}EWE}}_{\mathcal{A}}(x) = \Pr[\mathsf{IND\text{-}EWE}^1_{\mathcal{A}}(x) \to 1] - \Pr[\mathsf{IND\text{-}EWE}^0_{\mathcal{A}}(x) \to 1]$$

We say that WKEM *is extractable indistinguishable for a set X of instances x if for any probabilistic and polynomial-time* IND-EWE *adversary $\mathcal{A}$, there exists a probabilistic and polynomial-time extractor $\mathcal{E}$ such that for all $x \in X$, $\mathcal{E}(x)$ outputs a witness of x with probability at least $\mathsf{Adv}^{\mathsf{IND\text{-}EWE}}_{\mathcal{A}}(x)$ or at least $\frac{1}{2}$ up to a negligible term. More precisely,*

$$\forall x \in X \quad \Pr[R(x, \mathcal{E}(x))] \geq \min\left(\mathsf{Adv}^{\mathsf{IND\text{-}EWE}}_{\mathcal{A}}(x), \frac{1}{2}\right) - \mathsf{negl}(\lambda)$$

Note that if no witness exists for x, $\mathcal{E}(x)$ outputs a witness with null probability. Hence, it must be the case that $\mathsf{Adv}^{\mathsf{IND\text{-}EWE}}_{\mathcal{A}}(x) = \mathsf{negl}(\lambda)$. This property for all x without witness is actually the weaker (non-extractable) security notion of witness encryption [10].

Ideally, we would adopt this definition for the set X of all possible words. The reason why we introduce X is to avoid some "pathological" words making our construction insecure, which is a limitation of our construction. As pathological words are also characterized by an NP relation, X can be seen as a common subset of NP and coNP languages. Decryption requires a witness for x belonging to the NP language.

Chvojka et al. [3] requires $\Pr[R(x, \mathcal{E}(x))]$ to be "non-negligible" (without defining what this means). In our notion, we require more. Namely, we require extraction to be as effective as the attack.

The reason why the extractor extracts with probability "at least $\mathsf{Adv} - \mathsf{negl}$ or at least $\frac{1}{2}$" is that when $\mathsf{Adv}^{\mathsf{IND\text{-}EWE}}_{\mathcal{A}}$ is close to 1, we do not care if the extractor is not as good as the adversary, which could be unnecessarily hard to prove. We only care that it is either "substantially good" (i.e. at least $\frac{1}{2}$) or at least as good as $\mathcal{A}$. This will become necessary when we will need to amplify the probability of success of an extractor, as it will be the case in the proof of Th. 6.

Faonio et al. [9, Def.3] use instead $\Pr[R(x, \mathcal{E}(x))] \geq \frac{1}{2}\mathsf{Adv}^{\mathsf{IND\text{-}EWE}}_{\mathcal{A}}(x)$ which is probably more elegant but not tight.

We define extractable one-way security, which is a weaker security notion than extractable indistinguishability. Later, we show that an extractable one-way scheme can be transformed into an extractable indistinguishable one by generic transformation. Hence, we will be able to focus on making an extractable one-way WKEM.

Definition 4 (Extractable one-wayness). *Let $(\mathcal{K}_\lambda, \mathsf{Enc}, \mathsf{Dec})$ be a* WKEM *for R. Given an adversary $\mathcal{A}$, we define the following game:*

Game $\mathsf{OW\text{-}EWE}_{\mathcal{A}}(1^\lambda, x)$*:*

1: *(ROM only) pick a random function* H
2: $\mathsf{Enc}(1^\lambda, x) \to K, \mathsf{ct}$
3: $\mathcal{A}(x, \mathsf{ct}) \to h$
4: ***return*** $\mathbb{1}_{h=K}$

In the random oracle model (ROM), the game starts by selecting a random hash function H *and* Enc*,* Dec*, and* $\mathcal{A}$ *are provided a secure oracle access to* H*. We define the advantage of* $\mathcal{A}$ *by*

$$\mathsf{Adv}^{\mathsf{OW\text{-}EWE}}_{\mathcal{A}}(x) = \Pr[\mathsf{OW\text{-}EWE}_{\mathcal{A}}(x) \to 1]$$

We say that WKEM *is extractable one-way for a set* X *of instances* x *if for any probabilistic and polynomial-time* OW-EWE *adversary* $\mathcal{A}$*, there exists a probabilistic and polynomial-time extractor* $\mathcal{E}$ *such that for all* $x \in X$*,* $\mathcal{E}(x)$ *outputs a witness of* x *with probability at least* $\mathsf{Adv}^{\mathsf{OW\text{-}EWE}}_{\mathcal{A}}(x)$ *or at least* $\frac{1}{2}$ *up to a negligible term:*

$$\forall x \in X \quad \Pr[R(x, \mathcal{E}(x))] \geq \min\left(\mathsf{Adv}^{\mathsf{OW\text{-}EWE}}_{\mathcal{A}}(x), \frac{1}{2}\right) - \mathsf{negl}(\lambda)$$

As for IND-EWE, we observe that security implies $\mathsf{Adv}^{\mathsf{OW\text{-}EWE}}_{\mathcal{A}}(x) = \mathsf{negl}(\lambda)$ when x has no witness.

Indistinguishable implies one-way. As a warm-up, we show the easy result that extractable indistinguishable implies extractable one-way.

Theorem 5. *Let* $(\mathcal{K}_\lambda, \mathsf{Enc}, \mathsf{Dec})$ *be a* WKEM *for* R*. We assume that* $1/|\mathcal{K}_\lambda|$ *is negligible. If* WKEM *is extractable indistinguishable for a set* X *of instances* x*,* WKEM *is also extractable one-way for* X*.*

The proof is given in the full version of this paper.

Strongly secure from weakly secure transform. We now propose a generic WKEM transformation from OW-EWE-secure to IND-EWE-secure. The construction uses a random oracle. Let $\mathsf{WKEM}_0 = (\mathcal{K}^0_\lambda, \mathsf{Enc}_0, \mathsf{Dec}_0)$ be an OW-EWE-secure WKEM and H be a random oracle from $\mathcal{K}^0_\lambda$ to $\mathcal{K}_\lambda$. Our transformation is $\mathsf{WKEM} = (\mathcal{K}_\lambda, \mathsf{Enc}, \mathsf{Dec})$ as follows:

$\underline{\mathsf{Enc}(1^\lambda, x)}$:
1: $\mathsf{Enc}_0(1^\lambda, x) \to h, \mathsf{ct}$
2: $K \leftarrow H(h)$
3: **return** K, ct

$\underline{\mathsf{Dec}(\omega, \mathsf{ct})}$:
4: $\mathsf{Dec}_0(\omega, \mathsf{ct}) \to h$
5: $K \leftarrow H(h)$
6: **return** K

The intuition behind this transformation is the hardness of guessing an input to the random oracle H from the output.

Theorem 6. *If* WKEM_0 *is extractable one-way for a set* X *of instances, then* WKEM *from the above transformation is extractable indistinguishable for* X *in the random oracle model.*

The proof is given in the full version of this paper.

4 Hidden Group With Hashing

We define a new structure HiGH with correctness and security notions.

4.1 Definitions

We define the hidden group with hashing (HiGH) by some polynomially bounded algorithms.

Definition 7 (Hidden group with hashing). *A hidden group with hashing (HiGH) in dimension* d *consists of the following algorithms:*

- $\mathsf{Gen}(1^\lambda) \to \mathsf{pgp}, \mathsf{tgp}$*: A probabilistic polynomial-time algorithm which generates at random some public group parameters* pgp *and some trapdoor group parameters* tgp*.*
- $\mathsf{Hom}(\mathsf{tgp}, x) \to y$*: A deterministic polynomial-time algorithm which maps* $x \in \mathbb{Z}^d$ *to* y*. We denote by* G_{tgp} *the set of all* $\mathsf{Hom}(\mathsf{tgp}, x)$*, for* $x \in \mathbb{Z}^d$*. When it is clear from context, we omit* tgp *and write* $\mathsf{Hom}(x)$ *and* G*. Hence,* $y \in G$*.*
- $\mathsf{Mul}(\mathsf{pgp}, y, y') \to z$*: A deterministic polynomial-time algorithm which maps a pair* (y, y') *to a new element* z*. We denote by* $S_{\mathsf{pgp},\mathsf{tgp}}$ *the smallest superset of* G_{tgp} *which is stable by this operation. When it is clear from context, we omit* pgp *and write* $\mathsf{Mul}(y, y')$ *and* S*. Hence,* $y, y', z \in S$ *and* $G \subseteq S$*.*
- $\mathsf{Prehash}(\mathsf{pgp}, y) \to h$*: A deterministic polynomial-time algorithm which maps an element* $y \in S$ *to a "pre-hash"* h *which belongs to another domain.*[4] *When it is clear from context, we omit* pgp *and write* $\mathsf{Prehash}(y)$*.*

[4] We call h a *pre-hash* because our construction for a WKEM is extractable one-way and we need the additional construction of Th. 6 to hash h after pre-hash and get the key K.

We define by induction

$$\mathsf{Pow}(y_1,\dots,y_t,1^{a_1},\dots,1^{a_t}) = \mathsf{Mul}(\mathsf{Pow}(y_1,\dots,y_t,1^{a_1},\dots,1^{a_t-1}),y_t)$$

for $a_t > 0$ *and*

$$\mathsf{Pow}(y_1,\dots,y_{t-1},y_t,1^{a_1},\dots,1^{a_{t-1}},1^0) = \mathsf{Pow}(y_1,\dots,y_{t-1},1^{a_1},\dots,1^{a_{t-1}})$$

with $\mathsf{Pow}(y_1,\dots,y_t,1^0,\dots,1^0,1^1) = y_t$.

We write the a_i *inputs to* Pow *in unary to stress that the complexity is polynomial in terms of* $\sum_i a_i$.

For HiGH *to be correct, these algorithms must be such that*

- *they are all polynomially bounded;*
- *for all* $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp},\mathsf{tgp})$ *and* $y, y' \in G$, *if* $\mathsf{Prehash}(y) = \mathsf{Prehash}(y')$ *then* $y = y'$;
- *for all* $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp},\mathsf{tgp})$, $x, x' \in \mathbb{Z}^d$, *and* $y, y' \in S$ *if* $\mathsf{Prehash}(y) = \mathsf{Prehash}(\mathsf{Hom}(x))$ *and* $\mathsf{Prehash}(y') = \mathsf{Prehash}(\mathsf{Hom}(x'))$, *then*

$$\mathsf{Prehash}(\mathsf{Mul}(y,y')) = \mathsf{Prehash}(\mathsf{Hom}(x+x')) \tag{1}$$

The idea of the HiGH is that there is a hidden group in which elements have multiple representations but a unique pre-hash. We can use Mul to find a representation of the product of two represented factors. The computation is somewhat blind. The interface to Mul and Prehash is public. The interface also comes with a hidden group homomorphism Hom from $\mathbb{Z}^d$ which requires a trapdoor tgp. The additional property of HiGH which will play a role is that Mul can make representations grow so that computing an exponential of a large integer is not possible. Note that there is no interface to compute inverses. Our proposed instance based on RSA will also make it hard. It will consist of integers modulo a hidden number.

Lemma 8. *Given a* HiGH, *we have the following properties.*

1. *For all* $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp},\mathsf{tgp})$, *for all* t, $(x_1,\dots,x_t) \in (\mathbb{Z}^d)^t$, *and non-negative integers* $a_1,\dots,a_t$, *if* $y_i = \mathsf{Hom}(x_i)$, $i = 1,\dots,t$, *we have*

$$\mathsf{Prehash}\left(\mathsf{Pow}\left(y_1,\dots,y_t,1^{a_1},\dots,1^{a_t}\right)\right) = \mathsf{Prehash}(\mathsf{Hom}(a_1x_1+\cdots+a_tx_t))$$

2. $\mathsf{Prehash}(S) = \mathsf{Prehash}(G)$
3. *For any* $y \in S$, *there exists a unique* $z \in G$ *such that* $\mathsf{Prehash}(y) = \mathsf{Prehash}(z)$. *We call* z reduced *and we denote it by*

$$\mathit{Red}(y) = G \cap \mathsf{Prehash}^{-1}(\mathsf{Prehash}(y))$$

4. *The $*$ operation on G defined by $y * y' = \mathsf{Red}(\mathsf{Mul}(y, y'))$ makes G an Abelian group and* Hom *a surjective group homomorphism from $\mathbb{Z}^d$ to G. We denote by* Ker *the kernel of* Hom*.*

The proof is given in the full version of this paper.

For instance (with $d = 1$), for $x \in \mathbb{Z}$, we can define $\mathsf{tgp} = (\ell, k)$, $\mathsf{Hom}(x) = k^x \bmod \ell$, $\mathsf{Mul}(y, y') = y \times y'$ in $\mathbb{Z}$, $\mathsf{pgp} = (n, g)$, and $\mathsf{Prehash}(y) = g^y \bmod n$, where g has order ℓ in $\mathbb{Z}_n^*$ and n is an RSA modulus. We obtain $G = \langle k \rangle \subset \mathbb{Z}_\ell^*$ and S is the set of integers which factor in G. We have $\mathsf{Red}(y) = y \bmod \ell$. Here, Hom is from $\mathbb{Z}$ to $\mathbb{Z}$ and the hidden group is a cyclic subgroup of $\mathbb{Z}_\ell^*$.

Actually, we focus on cases where G is cyclic. As we will need representation of group elements to be hard to forge except by making generic use of Mul on known group elements, we will make G as a sparse subgroup of a supergroup $\bar{G}$.

For instance, for $x \in \mathbb{Z}$, we can define $\mathsf{tgp} = (\ell, k, k^\theta)$ for some invertible θ and $\mathsf{Hom}(x) = (k^x \bmod \ell, k^{\theta x} \bmod \ell)$ with the regular Mul in $\mathbb{Z}^2$. We obtain Hom from $\mathbb{Z}$ to $\mathbb{Z}^2$ and the hidden group is a cyclic subgroup of $(\mathbb{Z}_\ell^*)^2$. Our construction from Section 6 is based on this, with higher dimension.

4.2 HiGH Knowledge Exponent Assumption (HiGH-KE)

In the following definition, X denotes a set of tuples $(x_1, \ldots, x_t, \mathsf{aux})$ where aux could be anything, which could potentially give a clue to the adversary about the instance $(x_1, \ldots, x_t)$.

Definition 9. *A* HiGH *of dimension d satisfies the* HiGH *Knowledge Exponent Assumption for a set X if for any PPT algorithm $\mathcal{A}'$, there exists a PPT algorithm $\mathcal{E}'$ such that for all $x \in X$, the probability that the following game returns 1 is negligible:*

Game $\mathsf{HiGH\text{-}KE}(1^\lambda, x)$*:*

1: *parse* $x = (x_1, \ldots, x_t, \mathsf{aux})$
2: $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$ ▷ *this defines* $G = \mathsf{Hom}_{\mathsf{tgp}}(\mathbb{Z}^d)$
3: $y_i \leftarrow \mathsf{Hom}(\mathsf{tgp}, x_i)$, $i = 1, \ldots, t$
4: $\mathcal{A}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, \mathsf{aux}) \to h$
5: ***if*** $h \notin \mathsf{Prehash}(\mathsf{pgp}, G)$ ***then*** *abort*[5]
6: *set ρ to the random coins used by $\mathcal{A}'$*
7: $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, \mathsf{aux}, \rho) \to (1^{a_1}, \ldots, 1^{a_t})$ ▷ $a_i \in \mathbb{N}$

[5] We stress that this step may not be simulatable by a PPT algorithm.

8: ***if*** $\mathsf{Prehash}(\mathsf{pgp}, \mathsf{Hom}(\mathsf{tgp}, a_1x_1 + \cdots + a_tx_t)) = h$ ***then return*** *0*
9: ***return*** *1*

The point is that whenever the adversary succeeds to forge an element h of $\mathsf{Prehash}(G)$, the extractor, who has the same view (including ρ), should almost always manage to express it as the Prehash of a combination of the known pairs (x_i, y_i), with small coefficients a_i.[6] It means that only algorithms $\mathcal{A}'$ making Mul operations can forge valid prehashes h. The nice thing about this assumption is that it allows to get similar results as in the generic group model (i.e., to extract the combination) by remaining in the standard model.

This assumption combines the preimage awareness of Prehash and the knowledge-of-exponent assumption in G. By Prehash being *preimage aware*, we mean that whenever $\mathcal{A}$ succeeds to forge an element of $\mathsf{Prehash}(S)$ (which is also $\mathsf{Prehash}(G)$), then he must know some preimage in S.

4.3 HiGH Kernel Assumption (HiGH-Ker)

In the following definition, X denotes a set of tuples $(x_1, \ldots, x_t, \mathsf{aux})$ like in the previous definition.

Definition 10. *A* HiGH *satisfies the* HiGH *Kernel assumption for a set X of instances $x = (x_1, \ldots, x_t, \mathsf{aux})$ if for any PPT algorithm $\mathcal{A}''$, for any $x \in X$, the probability that the following game returns 1 is negligible.*

Game $\mathsf{HiGH\text{-}Ker}(1^\lambda, x)$*:*
1: *parse* $x = (x_1, \ldots, x_t, \mathsf{aux})$
2: $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$
3: $y_i \leftarrow \mathsf{Hom}(\mathsf{tgp}, x_i)$, $i = 1, \ldots, t$
4: *run* $\mathcal{A}''(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}) \to z$
5: ***if*** $z = 0$ ***then*** *abort*
6: ***if*** $\mathsf{Prehash}(\mathsf{pgp}, \mathsf{Hom}(\mathsf{tgp}, z)) \neq \mathsf{Prehash}(\mathsf{pgp}, \mathsf{Hom}(\mathsf{tgp}, 0))$ ***then*** *abort*
7: ***return*** 1

This means that even with a few (x_i, y_i) pairs for Hom, it is hard to find a kernel element.

5 WKEM from HiGH

We now construct a WKEM based on a HiGH and prove its security.

[6] a_i is small because it is retrieved in unary by a polynomially bounded algorithm $\mathcal{E}'$.

We abstract our WKEM scheme with HiGH for the Multi-SS language on Fig. 1. Essentially, to encrypt with $x = (x_1, \ldots, x_t, s)$, we generate a HiGH, we put all $y_i = \mathsf{Hom}(x_i)$ in the ciphertext, and the plaintext is

$$h = \mathsf{Prehash}(\mathsf{Hom}(s))$$

To decrypt with $\omega = (1^{a_1}, \ldots, 1^{a_t})$, we compute

$$h' = \mathsf{Prehash}(\mathsf{Pow}(y_1, \ldots, y_t, \omega)).$$

To present our construction in the frame of Barta et al. [2], we have a 2-message predictable argument for x in which the verifier generates from x a query $q = \mathsf{ct}$ and an expected response $\mathsf{st} = h$: $\mathcal{Q}(x) \to (q, \mathsf{st})$. The prover computes $\mathcal{P}(q, x, \omega) \to \pi$ the proof π which is accepted if $\pi = h$.

We first show that our construction is correct. Due to the correctness property (1) of HiGH, we have

$$\mathsf{Prehash}(\mathsf{Pow}(y_1, \ldots, y_t, \omega)) = \mathsf{Prehash}(\mathsf{Hom}(a_1 x_1 + \cdots + a_t x_t)).$$

If $R(x, \omega)$ holds, we have $a_1 x_1 + \cdots + a_t x_t = s$. We can then deduce that $h = h'$.

$\underline{\mathsf{Enc}(1^\lambda, x)}$:
1: parse $x = (x_1, \ldots, x_t, s)$
2: $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$
3: $y_i \leftarrow \mathsf{Hom}(\mathsf{tgp}, x_i)$, $i = 1, \ldots, t$
4: $z \leftarrow \mathsf{Hom}(\mathsf{tgp}, s)$
5: $h \leftarrow \mathsf{Prehash}(\mathsf{pgp}, z)$
6: set $\mathsf{ct} = (y_1, \ldots, y_t, \mathsf{pgp})$
7: **return** h, ct

$\underline{\mathsf{Dec}(\omega, \mathsf{ct})}$:
8: parse $\mathsf{ct} = (y_1, \ldots, y_t, \mathsf{pgp})$
9: parse $\omega = (1^{a_1}, \ldots, 1^{a_t})$
10: $z' \leftarrow \mathsf{Pow}(\mathsf{pgp}, y_1, \ldots, y_t, \omega)$
11: $h' \leftarrow \mathsf{Prehash}(\mathsf{pgp}, z')$
12: **return** h'

Fig. 1. WKEM construction

We show that WKEM is an extractable one-way witness key encapsulation mechanism for instances of Multi-SS.

Theorem 11. *The* WKEM *construction for* Multi-SS *on Fig. 1 is extractable one-way for a set X of instances $x = (x_1, \ldots, x_t, s)$ if the underlying* HiGH *satisfies the* HiGH-KE *and the* HiGH-Ker *assumptions for X.*

Proof. Let $\mathcal{A}$ be an OW-EWE adversary. We first construct an algorithm $\mathcal{A}'$ for the HiGH-KE game in Section 4.1 as follows which receives a target s as an auxiliary input aux:

$\mathcal{A}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, \mathsf{aux}; \rho)$:

1: parse s from aux
2: $x \leftarrow (x_1, \ldots, x_t, s)$
3: $\mathsf{ct} \leftarrow (y_1, \ldots, y_t, \mathsf{pgp})$
4: $\mathcal{A}(x, \mathsf{ct}; \rho) \to h$
5: **return** h

Thanks to the HiGH-KE assumption, there exists an extractor $\mathcal{E}'$ making the HiGH-KE game return 1 with negligible probability for every $x \in X$. We then construct the OW-EWE extractor $\mathcal{E}$ as follows:

$\mathcal{E}(1^\lambda, x, \mathsf{ct}, \rho)$:

1: parse $x = (x_1, \ldots, x_t, s)$
2: parse $\mathsf{ct} = (y_1, \ldots, y_t, \mathsf{pgp})$
3: set aux to s
4: $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, \mathsf{aux}, \rho) \to \omega$
5: **return** ω

Next, we will prove that $\mathcal{E}$ extracts well with nearly the same probability as $\mathcal{A}'$ wins in OW-EWE.

Below, we detail the OW-EWE game (on the left) and the HiGH-KE game (on the right). To make the comparison easier, we expanded Enc and $\mathcal{A}'$ in gray in a line starting with a dot.

$\mathsf{OW\text{-}EWE}(1^\lambda, x)$:

1: . parse $x = (x_1, \ldots, x_t, s)$
2: . $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$
3: . $y_i \leftarrow \mathsf{Hom}(x_i)$, $i = 1, \ldots, t$
4: . $z \leftarrow \mathsf{Hom}(s)$
5: . $K \leftarrow \mathsf{Prehash}(z)$
6: . $\mathsf{ct} \leftarrow (y_1, \ldots, y_t, \mathsf{pgp})$
7: $\mathcal{A}(x, \mathsf{ct}) \to h$
8: **return** $\mathbb{1}_{h=K}$

$\mathsf{HiGH\text{-}KE}(1^\lambda, x)$:

1: parse $x = (x_1, \ldots, x_t, s)$
2: $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$
3: $y_i \leftarrow \mathsf{Hom}(x_i)$, $i = 1, \ldots, t$
4: . $\mathsf{ct} \leftarrow (y_1, \ldots, y_t, \mathsf{pgp})$
5: . $\mathcal{A}(x, \mathsf{ct}; \rho) \to h$
6: **if** $h \notin \mathsf{Prehash}(G)$ **then** abort
7: set ρ to the random coins used by $\mathcal{A}$
8: $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, s, \rho) \to (1^{a_1}, \ldots, 1^{a_t})$
9: **if** $\mathsf{Prehash}(\mathsf{Hom}(a_1 x_1 + \cdots + a_t x_t)) = h$ **then return** 0
10: **return** 1

Clearly, everything until Step 6 is equivalent, with the same random coins. When OW-EWE returns 1, h is in $\mathsf{Prehash}(G)$ so HiGH-KE does not abort. Instead, HiGH-KE returns 0 or 1. We know by assumption that

HiGH-KE returns 1 with negligible probability. Hence,

$$\Pr[\mathsf{HiGH\text{-}KE} \to 0] \geq \Pr[\mathsf{OW\text{-}EWE} \to 1] - \mathsf{negl}$$

Cases when HiGH-KE returns 0 are the one when $\mathcal{E}'$ extracts successfully. Therefore, $\mathcal{E}'$ extracts $(1^{a_1}, \ldots, 1^{a_t})$ satisfying $\mathsf{Prehash}(\mathsf{Hom}(a_1x_1 + \cdots + a_tx_t)) = h$ with probability at least $\Pr[\mathsf{OW\text{-}EWE} \to 1] - \mathsf{negl}$. Due to the properties of HiGH, this implies that $a_1x_1 + \cdots + a_tx_t - s \in \mathsf{Ker}$.

We construct the following algorithm playing the HiGH-Ker game:

$\mathcal{A}''(x_1, \ldots, x_t, s, y_1, \ldots, y_t, z, \mathsf{pgp})$:
1: set $x = (x_1, \ldots, x_t, s)$
2: $\mathsf{ct} \leftarrow (y_1, \ldots, y_t, \mathsf{pgp})$
3: $\mathcal{A}(x, \mathsf{ct}) \to h$
4: set ρ to the random coins used by $\mathcal{A}$
5: $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, s, \rho) \to (1^{a_1}, \ldots, 1^{a_t})$
6: **return** $a_1x_1 + \cdots + a_tx_t - s$

Note that (s, z) plays the role of a new pair (x_{t+1}, y_{t+1}) here.

We now expand $\mathcal{A}''$ in the $HiGHKer$ game and we compare it to the HiGH-KE game:

HiGH-Ker$(1^\lambda, x)$:
1: parse $x = (x_1, \ldots, x_t, s)$
2: $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$
3: $y_i \leftarrow \mathsf{Hom}(x_i)$, $i = 1, \ldots, t$
4: . $\mathsf{ct} \leftarrow (y_1, \ldots, y_t, \mathsf{pgp})$
5: . $\mathcal{A}(x, \mathsf{ct}) \to h$
6: . set ρ to the random coins used by $\mathcal{A}$
7: . $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, s, \rho) \to (1^{a_1}, \ldots, 1^{a_t})$
8: . $z \leftarrow a_1x_1 + \cdots + a_tx_t - s$
9: **if** $z = 0$ **then** abort
10: **if** $\mathsf{Prehash}(\mathsf{pgp}, \mathsf{Hom}(\mathsf{tgp}, z)) \neq \mathsf{Prehash}(\mathsf{pgp}, \mathsf{Hom}(\mathsf{tgp}, 0))$ **then** abort
11: **return** 1

HiGH-KE$(1^\lambda, x)$:
1: parse $x = (x_1, \ldots, x_t, s)$
2: $\mathsf{Gen}(1^\lambda) \to (\mathsf{pgp}, \mathsf{tgp})$
3: $y_i \leftarrow \mathsf{Hom}(x_i)$, $i = 1, \ldots, t$
4: . $\mathsf{ct} \leftarrow (y_1, \ldots, y_t, \mathsf{pgp})$
5: . $\mathcal{A}(x, \mathsf{ct}; \rho) \to h$
6: **if** $h \notin \mathsf{Prehash}(G)$ **then** abort
7: set ρ to the random coins used by $\mathcal{A}$
8: $\mathcal{E}'(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}, s, \rho) \to (1^{a_1}, \ldots, 1^{a_t})$
9: **if** $\mathsf{Prehash}(\mathsf{Hom}(a_1x_1 + \cdots + a_tx_t)) = h$ **then return** 0
10: **return** 1

If HiGH-KE returns 0, then HiGH-Ker with the same coins either returns 1 (which happens with negligible probability) or $z = 0$. Hence, $\Pr[\mathsf{HiGH\text{-}Ker} \to z = 0] \geq \Pr[\mathsf{OW\text{-}EWE} \to 1] - \mathsf{negl}$. We deduce $\Pr[\mathcal{E}\ \mathsf{extracts}] \geq \Pr[\mathsf{OW\text{-}EWE} \to 1] - \mathsf{negl}$. □

Reusability of the parameters. The generated HiGH parameters ($\mathsf{pgp}, \mathsf{tgp}$) may require some computational effort in each encryption. This could be amortized by reusing some of the values. Namely, in our proposed HiGH, the parameters ℓ, n, g could be reused. Since generating the parameters $(k, \alpha_1, \ldots, \alpha_d, \theta)$ requires no effort, it is advised not to reuse them. Indeed, reusing them would help an adversary to pool many (x_i, y_i) pairs in the very same structure. It is also nice not to store them as the dimension d can be very large. Then, finding a linear relation with small coefficients would become easier and easier with the number of pairs.

However, using a long term ℓ is harming *forward secrecy* because disclosing it allows to decrypt all encryptions.

6 Our Instantiation of HiGH

6.1 Construction

We propose an instantiation of HiGH from the hardness of factorization of RSA modulus. Let $\mathsf{GenRSA}(1^\lambda)$ be an algorithm which outputs a tuple (n, ℓ, g) where n is an RSA modulus and g is an element of order ℓ in $\mathbb{Z}_n^*$. Our proposed instance is given in Fig. 2. Gen runs GenRSA and generates $k \in \mathbb{Z}_\ell^*$ so that g^k has order ℓ in $\mathbb{Z}_n^*$. The values k and ℓ are not revealed. They are used to derive a sequence $(k_1, k'_1, \ldots, k_d, k'_d)$ of elements of $\mathbb{Z}_\ell^*$ such that there is a hidden relation $k'_i = k_i^\theta \pmod{\ell}$ with $k_i = k^{\alpha_i} \pmod{\ell}$ for $i = 1, \ldots, d$.

The hidden group is $\bar{G} = (\mathbb{Z}_\ell^*)^2$ which has representation in $\mathbb{Z}^2$. Our homomorphism Hom goes from $\mathbb{Z}^d$ to $\bar{G} = (\mathbb{Z}_\ell^*)^2$ by $(\xi_1, \ldots, \xi_d) \mapsto (k_1^{\xi_1} \cdots k_d^{\xi_d}, {k'_1}^{\xi_1} \cdots {k'_d}^{\xi_d}) \bmod \ell$, but Mul treats $\bar{G}$ elements as belonging to $\mathbb{Z}^2$ and does the multiplication therein. When reduced modulo ℓ, we fall back to the hidden group. Hence, we use as Prehash the basis-g exponential modulo n because g has order ℓ. The exponential is made on the two components of the input from $\mathbb{Z}^2$.

G is the subgroup of $\bar{G}$ generated by (k, k^θ). We have $\mathsf{Red}(\nu_1, \nu_2) = (\nu_1 \bmod \ell, \nu_2 \bmod \ell)$. The kernel is a subgroup Ker of $\mathbb{Z}^d$ of all $(\xi_1, \ldots, \xi_d)$ such that $\alpha_1\xi_1 + \cdots + \alpha_d\xi_d = 0$ modulo the order of k. Prehash is injective when restricted on $\mathbb{Z}_\ell^2$, so it is injective when restricted on G. G is the hidden group of the super-structure $S \subseteq \mathbb{Z}^2$. We stress that Mul makes operations in the super-structure S of the hidden group G, and that Pow needs the a_i to be small because elements of $\mathbb{Z}^2$ can become huge when raised to the power a_i.

In our construction, the input of Hom is in $\mathbb{Z}^d$ as it comes from a vector in a Multi-SS instance. Contrarily, the output of Hom is in $\mathbb{Z}^2$. Having two

components allows to make it part of a sparse subset in which it is hard to forge elements without knowing a relation with known elements.

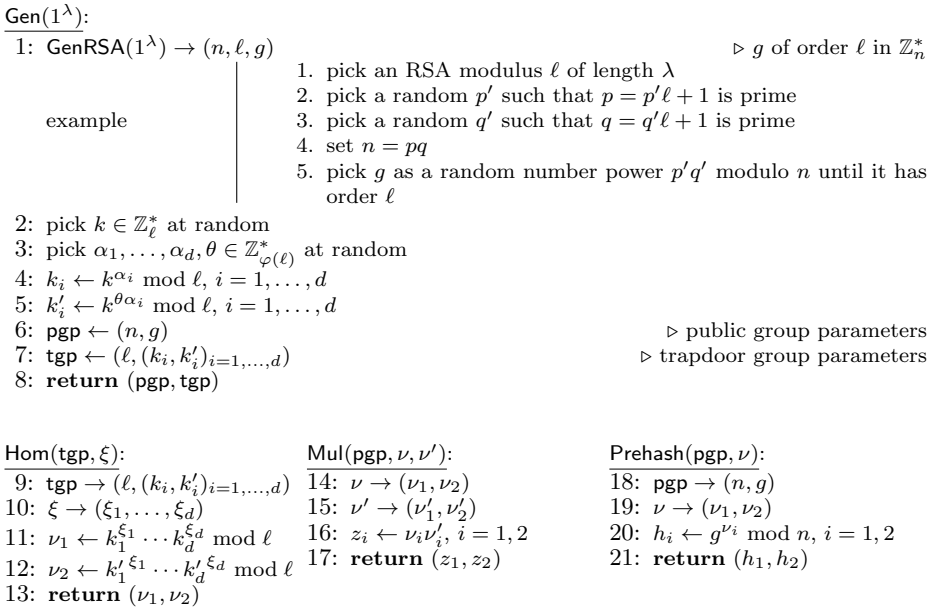

Fig. 2. Our HiGH Construction

Efficient implementation. To reduce the size of tgp, we can replace the $(k_i, k'_i)_i$ family by a seed which generates $k, \alpha_1, \ldots, \alpha_d, \theta$.

6.2 Possible Attacks

Finding a multiple ℓ' of ℓ allows an adversary to do multiplications in $\mathbb{Z}_{\ell'}^2$ which is a supergroup of G and to keep the size of numbers bounded by ℓ'. Hence, the adversary can implement some Mul′ and Pow′ algorithms which match to Mul and Pow when reduced modulo ℓ. Given enough known pairs $(x_i, \mathsf{Hom}(x_i))$, the adversary can find some integers a_i such that $a_1 x_1 + \cdots + a_t x_t = (0, \ldots, 0, 1, 0, \ldots, 0)$ over $\mathbb{Z}$. Using Pow′ allows to recover a pair equal to (k_j, k'_j) modulo ℓ'. Next, the adversary can implement an $\xi \mapsto \mathsf{Hom}'(\xi)$ algorithm which matches Hom when reduced modulo ℓ.

We can then make a HiGH-KE adversary who picks some random (large enough, with polynomial length) ξ, then $h = \mathsf{Prehash}(\mathsf{Hom}'(\xi))$.

Either it breaks the HiGH-KE assumption, or there exists an extractor who extracts some (polynomially bounded) $a_1, \ldots, a_t$ such that $h = \mathsf{Prehash}(\mathsf{Hom}'(\xi'))$ with $\xi' = a_1 x_1 + \cdots + a_t x_t$. In the latter case, we let $z = \xi - \xi'$. Since the set of all possible ξ' is polynomially bounded while the set of all possible ξ has exponential size, $z = 0$ happens with negligible probability. Hence, the adversary producing z breaks the HiGH-Ker assumption. Therefore, either HiGH-KE or HiGH-Ker is broken.

Essentially, the adversary recovers some tgp' which is functionally equivalent to tgp and allows to break the scheme. For this reason, it is essential that the adversary cannot find a relation which is true modulo ℓ but not in $\mathbb{Z}$, as it would reveal a multiple of ℓ. We give an example below.

Small solutions to the homogeneous problem. Given some (x_i, y_i) pairs with $y_i = \mathsf{Hom}(x_i)$, if an adversary finds some small[7] a'_i (positive or negative) such that $\sum_i a'_i x_i = 0$, then he can compute

$$\ell'_j = \prod_i y_{i,j}^{\max(0,a'_i)} - \prod_i y_{i,j}^{\max(0,-a'_i)}$$

for $j = 1, 2$, which are multiples of ℓ. Their gcd ℓ' is a multiple of ℓ which can be used in its place. Then, the adversary can break the HiGH.

This means that for the HiGH-Ker to hold on X, there should be no $x \in X$ with a small relation $\sum_i a'_i x_i = 0$. This is why we must consider a subset X of all instances of Multi-SS.

6.3 Security Results

Knowledge-of-exponent assumption. Wu and Stinson define the generalized knowledge-of-exponent assumption (GKEA) [16] over a group G. It says that for an adversary who gets $y_1, \ldots, y_t \in G$ and succeeds to produce $z \in G$, there must be an extractor who would, with the same view, make $a_1, \ldots, a_t \in \mathbb{Z}$ such that $z = y_1^{a_1} \times \cdots \times y_t^{a_t}$. In our group $G = \langle (k, k^\theta) \rangle \subset (\mathbf{Z}_\ell^*)^2$, this assumption is usual, even when ℓ is known. In our settings, ℓ is not known but a preimage x_i by Hom for each y_i is known.

We conjecture that the HiGH-KE assumption holds in our construction for every $(x_1, \ldots, x_t, s)$ such that there is no relation $a_1 x_1 + \cdots + a_t x_t = 0$ with small a_i.

[7] By "small", we mean that computing $y_i^{|a'_i|}$ over $\mathbb{Z}$ is doable.

Kernel assumption. The *RSA order assumption* says that given an RSA modulus ℓ and a random $k \in \mathbb{Z}_\ell^*$, it is hard to find a positive integer z' such that $k^{z'} \bmod \ell = 1$. The game is defined relative to the distribution P of ℓ as follows:

1: pick a random ℓ following P
2: pick a random $k \in \mathbb{Z}_\ell^*$ uniformly
3: run $\mathcal{B}(\ell, k) \to z'$
4: **if** $z' = 0$ or $k^{z'} \bmod \ell \neq 1$ **then** abort
5: **return** 1

For our construction, we can prove that the HiGH-Ker problem is at least as hard as the RSA order problem when ℓ is a strong RSA modulus. However, the HiGH-Ker problem is likely to be hard even when ℓ is not a strong RSA modulus.

Theorem 12. *We assume that* GenRSA *generates only strong RSA moduli ℓ and we let P be their distribution. Given an adversary $\mathcal{A}$ with advantage $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{HiGH\text{-}Ker}}$ in the $\mathsf{HiGH\text{-}Ker}(1^\lambda, x)$ game for a given x, we can construct an adversary $\mathcal{B}$ with same advantage (up to a negligible term) in the order problem with this modulus distribution, and similar complexity.*

Proof. We consider an adversary $\mathcal{A}$ playing the HiGH-Ker game with input x. We define an adversary $\mathcal{B}(\ell, k)$ playing the order game. The adversary $\mathcal{B}$ receives (ℓ, k) from the order game then simulates the rest of the HiGH-Ker game with $\mathcal{A}$. This simulator must generate n. There is a little problem to select $\alpha_1, \ldots, \alpha_d, \theta$ because $\mathcal{B}$ does not know $\varphi(\ell)$. However, if ℓ is a strong RSA modulus, by sampling in a domain which is large enough and with only odd integers, the statistical distance Δ between the real and simulated distributions of $(\mathsf{pgp}, \mathsf{tgp})$ is negligible. $\mathcal{A}$ may give some kernel elements $z = (z_1, \ldots, z_d)$ from which $\mathcal{B}$ can compute $\alpha_1 z_1 + \cdots + \alpha_d z_d$.

More precisely, $\mathcal{B}(\ell, k)$ works as follows:

$\mathcal{B}(\ell, k)$:
1: pick a random p' such that $p = p'\ell + 1$ is prime
2: pick a random q' such that $q = q'\ell + 1$ is prime
3: set $n = pq$
4: pick $g$as a random number power $p'q'$ modulo n until $g \neq 1$
▷ g has order ℓ except with negligible probability
5: pick $\alpha_1, \ldots, \alpha_d, \theta$ odd in $\{0, \ldots, B-1\}$, $B = \ell^2$
▷ statistical distance with correct $(\alpha_1, \ldots, \alpha_d, \theta)$ is $\Delta = \mathsf{negl}(\lambda)$
6: $k_1 \leftarrow k^{\alpha_i} \bmod \ell$, $i = 1, \ldots, d$
7: $k_1' \leftarrow k^{\theta\alpha_i} \bmod \ell$, $i = 1, \ldots, d$
8: $\mathsf{pgp} \leftarrow (n, g)$

9: $\mathsf{tgp} \leftarrow (\ell, (k_i, k'_i)_{i=1,\ldots,d})$
10: $y_i \leftarrow \mathsf{Hom}(\mathsf{tgp}, x_i)$, $i = 1, \ldots, d$
11: run $\mathcal{A}(x_1, \ldots, x_t, y_1, \ldots, y_t, \mathsf{pgp}) \to z$
12: **return** z

The two steps which deviate from the HiGH-Ker game played by $\mathcal{A}$ introduce no noticeable difference. Namely, it is rare that g does not have order g (it is similar than finding factors of ℓ by chance). The statistical distance Δ is negligible, as estimated by the lemma below. Hence, $\mathcal{B}$ succeeds in his game with (nearly) the same probability as $\mathcal{A}$ succeeds in the HiGH-Ker game. □

Lemma 13. *If $\ell = \ell_p \times \ell_q$ is a strong RSA modulus, then the statistical distance Δ between the modulo $\varphi(\ell)$ reduction of the two following generators*

1. *pick* $(\alpha_1, \ldots, \alpha_d, \theta) \in \left(\mathbf{Z}^*_{\varphi(\ell)}\right)^{d+1}$ *uniformly at random*
2. *pick* $(\alpha_1, \ldots, \alpha_d, \theta) \in \{1, 3, 5, \ldots, B-2\}^{d+1}$ *uniformly at random*

where $B = \ell^2$ is such that

$$\Delta \leq (d+1)\left(\frac{2}{\ell_p - 1} + \frac{2}{\ell_q - 1} + \frac{\ell}{B-1}\right)$$

Note that the second generator simply picks odd elements in $\{0, \ldots, B-1\}$.

Proof. The two generators take independent $d+1$ samples from two distributions. Hence, Δ is bounded by $d+1$ times the statistical distance between these two distributions.

The first distribution is uniform in $\mathbf{Z}^*_{\varphi(\ell)}$.

The second distribution selects an element in $\{1, 3, \ldots, B-1\}$ then reduces it modulo $\varphi(\ell)$.

We define an intermediate distribution which uniform in $\{1, 3, \ldots, \varphi(\ell)-1\}$, which is a superset of $\mathbf{Z}^*_{\varphi(\ell)}$.

Clearly, the statistical distance between the first distribution and the intermediate distribution is bounded by the probability to take a multiple of $\ell_{p'}$ or a multiple of $\ell_{q'}$, which is bounded by the sum $\frac{2}{\ell_p - 1} + \frac{2}{\ell_q - 1}$.

The statistical distance between the intermediate distribution and the second distribution is bounded by the cardinality $\varphi(\varphi(\ell))$ times the gap between the largest and the lowest probabilities in the distribution, which is $\frac{2}{B-1}$. This is

$$2\frac{\varphi(\varphi(\ell))}{B-1} \leq \frac{\ell}{B-1}$$

□

7 Conclusion

We have shown how to construct a WKEM for a variant of the subset sum problem, based on HiGH. This is secure in the generic HiGH model. We proposed an HiGH construction based on RSA which has a restriction on the subset sum instances. One open question is to make it work even for instances having a small linear combination which vanishes.

Another interesting challenge is to build a post-quantum HiGH.

Acknowledgement

Gwangbae Choi is supported by the Swiss National Science Foundation (SNSF) Project funding no. 169110.

References

1. Abe, M., Fehr, S.: Perfect NIZK with adaptive soundness. In: Theory of Cryptography Conference. pp. 118–136. Springer (2007)
2. Barta, O., Ishai, Y., Ostrovsky, R., Wu, D.J.: On succinct arguments and witness encryption from groups. In: Advances in Cryptology - CRYPTO 2020. Lecture Notes in Computer Science, vol. 12170, pp. 776–806. Springer Berlin Heidelberg (2020), also IACR Eprint 2020/1319
3. Chvojka, P., Jager, T., Kakvi, S.A.: Offline witness encryption with semi-adaptive security. In: Applied Cryptography and Network Security - ACNS 2020. pp. 231–250. Springer (2020)
4. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Annual International Cryptology Conference. pp. 13–25. Springer (1998)
5. Cramer, R., Shoup, V.: Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 45–64. Springer (2002)
6. Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM Journal on Computing **33**(1), 167–226 (2003)
7. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Annual International Cryptology Conference. pp. 445–456. Springer (1991)
8. Dent, A.W.: The hardness of the DHK problem in the generic group model. IACR Cryptology ePrint Archive **2006**, 156 (2006)
9. Faonio, A., Nielsen, J.B., Venturi, D.: Predictable arguments of knowledge. In: Public-Key Cryptography - PKC 2017. Lecture Notes in Computer Science, vol. 10174, pp. 121–150. Springer Berlin Heidelberg (2017), also IACR Eprint 2015/740
10. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: Proceedings of the forty-fifth annual ACM symposium on Theory of computing. pp. 467–476. ACM (2013)

11. Goldwasser, S., Kalai, Y.T., Popa, R.A., Vaikuntanathan, V., Zeldovich, N.: How to run Turing machines on encrypted data. In: Annual Cryptology Conference. pp. 536–553. Springer (2013)
12. Groth, J., Ostrovsky, R., Sahai, A.: Perfect non-interactive zero knowledge for np. In: Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 339–358. Springer Berlin Heidelberg (2006), also IACR Eprint 2005/290
13. Karp, R.M.: Reducibility among combinatorial problems. In: Complexity of computer computations, pp. 85–103. Springer (1972)
14. Liu, J., Jager, T., Kakvi, S.A., Warinschi, B.: How to build time-lock encryption. Designs, Codes and Cryptography **86**(11), 2549–2586 (2018)
15. Naor, M.: On cryptographic assumptions and challenges. In: Annual International Cryptology Conference. pp. 96–109. Springer (2003)
16. Wu, J., Stinson, D.R.: An efficient identification protocol and the knowledge-of-exponent assumption. IACR Cryptology ePrint Archive **2007**, 479 (2007)