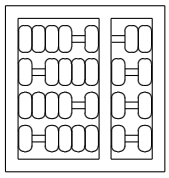

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Ideal Lattice-based (H)IBE Scheme

Karina Mochetti *Ricardo Dahab*

Technical Report - IC-14-18 - Relatório Técnico

November - 2014 - Novembro

Ideal Lattice-based (H)IBE Scheme

Karina Mochetti Ricardo Dahab*

Abstract

In an Identity-Based Encryption Scheme, the public key is based on unique user's information, such as an email address, so it is possible to perform the decryption without public-key certification. Ideal lattices are a generalisation of cyclic lattices, in which the lattice corresponds to ideals in a ring. They allow some improvements in the parameters size and multiplication complexity. In this paper, we present a version of the lattice-based (H)IBE scheme by Agrawal, Boneh, Boyen (Eurocrypt'10) for ideal lattices. As the underlying (H)IBE scheme, our scheme is shown to be weak selective secure based on the difficulty of the Learning With Errors Problem (LWE), but our new primitive has smaller public keys.

1 Introduction

Functional encryption has become increasingly important over the years because it provides users with a much finer control of decryption capabilities. More specifically, in a functional encryption system secret keys allow users to learn functions of encrypted data, i.e., for a message m and a value k it is possible to evaluate a function $f(k, m)$ given the encryption of m and a secret key sk_k. Some examples of functional encryption are Identity-Based Encryption (IBE) [5, 7], Attribute-Based Encryption (ABE) [20, 9] and Inner-Product Encryption (IPE) [10, 15].

In this paper, we focus on the notion of Identity-Based Encryption (IBE), that was proposed by Shamir [21] in 1984, but remained as an open problem until 2001 when Boneh and Franklin [5] and Cocks [7] constructed the first schemes. In an IBE scheme the public-key is based on unique user's information, such as an email address, so it is possible to perform the decryption without the need for public-key certification.

Boneh and Franklin's IBE scheme is based on bilinear pairings, while Cock's IBE scheme is based on the quadratic residuosity problem. The first lattice-based IBE scheme was proposed by Gentry et al. [8] and its security is based on the LWE problem in the random oracle model. Another lattice-based IBE scheme, but with hierarchical property, was proposed by Agrawal et al. [1] and it is based on the Bonsai Trees concept [8, 6]; it does not use random oracles, but it is also based on the LWE problem.

For cryptosystems, such as functional schemes, that use a Private Key Generator (PKG), it is convenient to have a hierarchy of certificate authorities, that is, the root certificate

*Institute of Computing, UNICAMP, Brazil

authority can issue certificates for other certificate authorities, which can issue certificates for users. For IBE schemes; a PKG have to compute private keys only to the entities immediately below them, this scheme is called hierarchical.

Ideal lattices are a generalisation of cyclic lattices, in which the lattice corresponds to ideals in a ring $\mathbb{Z}[x]/\langle f(x)\rangle$, for some irreducible polynomial function f. They can be used to decrease the parameters needed to describe a lattice and its basis pattern can be used to improve the matrix multiplication complexity. Cyclic lattices were presented by Micciancio [13] along with the first provably secure one-way function based on the worst-case hardness of the restriction of poly(n)-SVP to cyclic lattices.

Based on the LWE problem, a ring-LWE version was defined in a wide class of rings and its hardness proved under worst-case assumptions on ideal lattices in these rings. This allowed the construction of collision-resistant hash functions [11, 18]. Later, a new variant of the LWE problem was defined, ideal-LWE [22], which was used to build an efficient public-key encryption scheme based on the worst-case hardness of the approximate SVP in ideal lattices.

An ideal lattice IBE scheme is presented by Okuhata et al. [16]. They use the ideal-LWE variant of the LWE problem to build an IBE scheme similar to the one presented by Agrawal et al. [1]. Their scheme has a fixed, therefore smaller, key size for all message sizes, while on the original paper the key size increases with the size of the message.

Another ideal lattice IBE scheme is presented by Yang et al. [23]. Although really similar, our schemes were developed independently of their scheme. Beside the ideal lattice IBE, we also provide the hierarchical ideal lattice version of the scheme.

2 Definitions

In this section we present the basic concepts used in our construction. Section 2.1 defines an identity-based encryption scheme and an hierarchical version, Section 2.2 gives definitions and properties of lattices, sampling algorithms (for general and ideal lattices) and the LWE and ring-LWE problems.

Notation

For any integer $q \geq 2$, we let $\mathbb{Z}_q$ denote the ring of integers modulo q and we represent $\mathbb{Z}_q$ as integers in $(q/2, q/2]$. We let $\mathbb{Z}_q^{n\times m}$ denote the set of $n \times m$ matrices with entries in $\mathbb{Z}_q$. We use capital letters (e.g. A) to denote matrices, bold lowercase letters (e.g. $\boldsymbol{v}$) to denote vectors. The notation $A^\top$ denotes the transpose of the matrix A. When we say a matrix defined over $\mathbb{Z}_q$ has *full rank*, we mean that it has full rank modulo each prime factor of q. If A_1 is an $n \times m$ matrix and A_2 is an $n \times m'$ matrix, then $[A_1\|A_2]$ denotes the $n \times (m + m')$ matrix formed by concatenating A_1 and A_2. If $\boldsymbol{w}_1$ is a length m vector and $\boldsymbol{w}_2$ is a length m' vector, then we let $[\boldsymbol{w}_1|\boldsymbol{w}_2]$ denote the length $(m + m')$ vector formed by concatenating $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$. However, when doing matrix-vector multiplication we always view vectors as column vectors. For a vector $\boldsymbol{v}$ we define as $|\boldsymbol{v}| = \sqrt{\sum \boldsymbol{x}_i^2}$ as the norm of vector $\boldsymbol{v}$ and for a matrix A, we define as $|A| = \max|A\boldsymbol{x}|$, for $|\boldsymbol{x}| = 1$, the norm of matrix A. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use negl(n) to

denote a negligible function of n. We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use poly(n) to denote a polynomial function of n. We say an event occurs with *overwhelming probability* if its probability is $1 - \text{negl}(n)$. Given a polynomial $f(x)$ we say a ring $\mathbb{Z}[x]/\langle f(x)\rangle$ is the set of all polynomials $g(x) \mod f(x)$ with coefficients in $\mathbb{Z}$. The notation $g(x) \otimes h(x)$ denotes the multiplication of polynomials $g(x)$ and $h(x) \in \mathbb{Z}[x]/\langle f(x)\rangle$ modulo $f(x)$.

2.1 Identity Based Encryption

Based on the definition of predicate encryption scheme by Katz et al. [10], we have that an Identity-Based Encryption Scheme consists of the following four algorithms:

$\mathsf{SetUp}(1^\lambda)$: Takes as input security parameter λ and outputs public-key mpk and master secret key msk.

$\mathsf{KeyGen}(mpk, msk, id)$: Takes as input public-key mpk, master secret key msk and an identity id and outputs a secret key sk.

$\mathsf{Enc}(mpk, msg, id)$: Takes as input msg in some associated message space, public-key mpk, an identity id and outputs a ciphertext ct.

$\mathsf{Dec}(mpk, ct, sk)$: Takes as input public-key mpk, ciphertext ct, secret key sk and outputs the message msg.

We make the following consistency requirement. Suppose ciphertext ct is obtained by running Enc on input public key mpk, message msg and identity id and that sk is a secret key for identity id' obtained through a call of KeyGen using the same mpk. Then Dec, on input mpk, ct and sk returns msg, except with negligible probability, if and only if $id = id'$.

In a hierarchical scheme, a user in level t can use his/her secret key to derive a secret key for a user at level $t + 1$. We have that an Hierarchical Identity-Based Encryption Scheme consists of the following four algorithms:

$\mathsf{SetUp}(1^\lambda, 1^\mu)$: Takes as input security parameter λ and an hierarchical format μ and outputs public-key mpk and master secret key msk, i.e., the secret key for level 0 ($sk_0 = msk$).

$\mathsf{KeyDerive}(mpk, sk_{t-1}, id)$: Takes as input public-key mpk, secret key sk_{t-1} for level $t-1$ and an identity $id = \{id_1, \cdots, id_t\}$ and outputs a secret key sk_t for level t.

$\mathsf{Enc}(mpk, msg, , id)$: Takes as input msg in some associated message space, public-key mpk, an identity $id = \{id_1, \cdots, id_t\}$ and outputs a ciphertext ct.

$\mathsf{Dec}(mpk, ct, sk_t)$: Takes as input public-key mpk, ciphertext ct, secret key sk_t for level t and outputs the message msg.

We make the following consistency requirement. Suppose ciphertext ct is obtained by running Enc on input public key mpk, message msg and identity $id = \{id_1, \cdots, id_t\}$ and that sk_t is a secret key for identity $id' = \{id'_1, \cdots, id'_t\}$ obtained through a call of $\mathsf{KeyDerive}$ using the same mpk. Then Dec, on input mpk, ct and sk_t returns msg, except with negligible probability, if and only if $id_i = id'_i$ for all $i \in [1, t]$.

Security is modelled by means of a game between a challenger $\mathcal{C}$ and a probabilistic polynomial-time adversary $\mathcal{A}$. In this work, we achieve *selective identity* security, meaning that $\mathcal{A}$ must declare its *challenge identities* before seeing the public-key.

Init: $\mathcal{A}$ outputs challenge identities $id_0^\star, id_1^\star$.

Setup: The challenger $\mathcal{C}$ runs the SetUp algorithm to generate public-keys mpk which it gives to the adversary $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ is given oracle access to $\mathsf{KeyGen}(mpk, msk, \cdot)$.

Challenge: The adversary $\mathcal{A}$ gives a pair of messages (msg_0, msg_1) to the challenger $\mathcal{C}$. Then $\mathcal{C}$ chooses random $\eta \xleftarrow{\$} \{0,1\}$, encrypts msg_η under id_η and sends the resulting ciphertext to $\mathcal{A}$.

Phase 2: The same as Phase 1.

Guess: The challenger $\mathcal{A}$ must output a guess η' for η.

If the advantage of every probabilistic polynomial time adversary $\mathcal{A}$ is defined to be $|\text{Prob}[\eta' = \eta] - \frac{1}{2}|$, then we say the scheme has indistinguishability under chosen-plaintext attack under the selective model (IND-sID-CPA for short).

2.2 Lattices

This section presents the collection of results from [2, 11, 22, 1, 14, 8, 6, 4] that we will need for our construction and proof of security.

An m-dimensional lattice Λ is a full-rank discrete subgroup of $\mathbb{R}^m$. A basis of Λ is a linearly independent set of vectors whose span is Λ. We will focus on integer lattices and among these we will focus on the q-ary lattices defined as follows: for any integer $q \geq 2$ and any $A \in \mathbb{Z}_q^{n \times m}$, we define

$$\begin{aligned}
\Lambda_q^{\perp}(A) &\leftarrow \{\boldsymbol{e} \in \mathbb{Z}^m : A\boldsymbol{e} = 0 \mod q\} \\
\Lambda_q^{\boldsymbol{u}}(A) &\leftarrow \{\boldsymbol{e} \in \mathbb{Z}^m : A\boldsymbol{e} = \boldsymbol{u} \mod q\} \\
\Lambda_q(A) &\leftarrow \{\boldsymbol{e} \in \mathbb{Z}^m : \exists\ \boldsymbol{s} \in \mathbb{Z}_q^m \text{ with } A^\top \boldsymbol{s} = \boldsymbol{e} \mod q\}.
\end{aligned}$$

Let $\mathcal{I}$ be an ideal of the ring $\mathcal{R} = \mathbb{Z}[x]/\langle f(x)\rangle$, i.e., a subset of $\mathcal{R}$ that is closed under addition and multiplication. The ideal $\mathcal{I}$ is a sublattice of $\mathbb{Z}^n$. For a ring $\mathcal{R} = \mathbb{Z}[x]/\langle f(x)\rangle$ we can define the basis of the ideal lattice $\Lambda_q^{\perp}(A)$, with $A = \mathsf{rot}_f(g) \in \mathbb{Z}_q^{n \times n}$, where each row i of A is given by the coefficients of $x^i g(x) \mod f(x)$ for $i \in \{0, n-1\}$.

For a polynomial $g(x) \in \mathcal{R}$, we can represent it as a vector $\boldsymbol{a}$ where for each $i \in \{0, n-1\}$, a_i is the coefficient of x^i in $g(x)$. We assume that any polynomial is a vector, and $\boldsymbol{a} \otimes \boldsymbol{b}$ is the multiplication of the polynomials represented by vectors $\boldsymbol{a}$ and $\boldsymbol{b}$. For a ring $\mathcal{R}$, we have that $\hat{\boldsymbol{g}} \in \mathcal{R}^k$ is a vector of k polynomials in $\mathcal{R}$. Since polynomials are easily represented as vectors, we denote by $\hat{\boldsymbol{v}}$ any concatenation of vectors, i.e., $\hat{\boldsymbol{v}} = [\boldsymbol{v}_0 | \ldots | \boldsymbol{v}_k]$, with $\boldsymbol{v}_i$ a vector.

Note that if $f(x) = x^n + 1$, then the matrix $A = \mathsf{rot}_f(g) \in \mathbb{Z}_q^{n \times n}$ is an anti-circulant matrix and for $f(x) = x^n - 1$, we have that the matrix $A = \mathsf{rot}_f(g) \in \mathbb{Z}_q^{n \times n}$ is a circulant matrix. These lattices are called cyclic lattices and they are a special class of ideal lattices.

Lemma 1. *For two matrices A and B, if A and B are bases for ideal lattices, then $C = [A|B]$ is also a basis for an ideal lattice.*

Lemma 2. *For $\boldsymbol{a}$ and $\boldsymbol{b} \in \mathcal{R}$ we have that $\mathsf{rot}_f(\boldsymbol{a}) + \mathsf{rot}_f(\boldsymbol{b}) = \mathsf{rot}_f(\boldsymbol{a} + \boldsymbol{b})$ and $\mathsf{rot}_f(\boldsymbol{a})\mathsf{rot}_f(\boldsymbol{b}) = \mathsf{rot}_f(\boldsymbol{a} \otimes \boldsymbol{b})$.*

Lemma 3. *([13, Lemma 4.4]) Let $\hat{\boldsymbol{b}} \in \mathcal{R}^k$ be a sequence of arbitrary ring elements. If $\hat{\boldsymbol{a}} \in \mathcal{R}^k$ are independently and uniformly distributed ring elements, then $\sum \boldsymbol{a}_i \otimes \boldsymbol{b}_i$ is uniformly distributed over the ideal generated by $\hat{\boldsymbol{b}}$. Note that for $k = 1$ we have that $\boldsymbol{a} \otimes \boldsymbol{b}$ is uniformly distributed for $\boldsymbol{a} \in \mathcal{R}$ and $\boldsymbol{b} \in \mathcal{R}$.*

The two main advantages of using ideal lattices are: The basis matrix $n \times m$ can be built from a polynomial with degree m, which results in smaller key sizes. The multiplication of a matrix that is a basis for an ideal lattice by a vector can be done in an efficient way [17].

For simplicity, we define two auxiliary functions. The first is called $\mathsf{Exp}()$, and it takes a vector $\hat{\boldsymbol{a}} \in \mathcal{R}^k$ and expands it to a matrix as follows:

$$\mathsf{Exp}(\hat{\boldsymbol{a}}) = \begin{bmatrix} \mathsf{rot}_f(\boldsymbol{a}') & O & \dots & O \\ O & \mathsf{rot}_f(\boldsymbol{a}_1) & \dots & O \\ \vdots & \vdots & \dots & \vdots \\ O & O & \dots & \mathsf{rot}_f(\boldsymbol{a}_{k-1}) \end{bmatrix}$$

The second is the $\mathsf{Rot}()$ function, that takes a vector $\hat{\boldsymbol{a}} \in \mathcal{R}^k$ and also expands it to a matrix as follows:

$$\mathsf{Rot}(\hat{\boldsymbol{b}}) = \left[\ \mathsf{rot}_f(\boldsymbol{b}_0) | \mathsf{rot}_f(\boldsymbol{b}_1) | \dots | \mathsf{rot}_f(\boldsymbol{b}_{k-1}) \ \right]$$

Note that

$$\mathsf{Rot}(\hat{\boldsymbol{b}})\mathsf{Exp}(\hat{\boldsymbol{a}}) = \left[\ \mathsf{rot}_f(\boldsymbol{a}')\mathsf{rot}_f(\boldsymbol{b}_0) | \dots | \mathsf{rot}_f(\boldsymbol{a}_{k-1})\mathsf{rot}_f(\boldsymbol{b}_{k-1}) \ \right]$$

Gram-Schmidt norm Let $S = \{\boldsymbol{s}_1, \dots, \boldsymbol{s}_k\}$ be a set of vectors in $\mathbb{R}^m$. Then, $\widetilde{S} \leftarrow \widetilde{\boldsymbol{s}}_1, \dots, \widetilde{\boldsymbol{s}}_k \subset \mathbb{R}^m$ denotes the *Gram-Schmidt orthogonalization* of the vectors $\boldsymbol{s}_1, \dots, \boldsymbol{s}_k$. We refer to $|\widetilde{S}|$ as the *Gram-Schmidt norm* of S.

Gaussian distributions Let L be a discrete subset of $\mathbb{Z}^n$. For any vector $\boldsymbol{c} \in \mathbb{R}^n$ and any positive parameter $\sigma \in \mathbb{R}_{>0}$, let $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w}) \leftarrow \exp\left(-\pi|x - \boldsymbol{c}|^2/\sigma^2\right)$ be the Gaussian function on $\mathbb{R}^n$ with center $\boldsymbol{c}$ and parameter σ. Let $\rho_{\sigma,\boldsymbol{c}}(L) \leftarrow \sum_{\boldsymbol{w} \in L} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w})$ be the discrete integral of $\rho_{\sigma,\boldsymbol{c}}$ over L, and let $\mathcal{D}_{L,\sigma,\boldsymbol{c}}$ be the discrete Gaussian distribution over L with center $\boldsymbol{c}$ and parameter σ. Specifically, for all $\boldsymbol{v} \in L$, we have $\mathcal{D}_{L,\sigma,\boldsymbol{c}}(\boldsymbol{v}) = \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{v})}{\rho_{\sigma,\boldsymbol{c}}(L)}$. For notational convenience, $\rho_{\sigma,0}$ and $\mathcal{D}_{L,\sigma,0}$ are abbreviated as ρ_σ and $\mathcal{D}_{L,\sigma}$ respectively.

The following lemma captures standard properties of these distributions.

Lemma 4. *Let $q \geq 2$ and let A be a matrix in $\mathbb{Z}_q^{n \times m}$ with $m > n$. Let T_A be a basis for $\Lambda_q^\perp(A)$ and $\sigma \geq |\widetilde{T_A}| \cdot \omega(\sqrt{\log m})$. Then, for $\boldsymbol{c} \in \mathbb{R}^m$ and $\boldsymbol{u} \in \mathbb{Z}_q^n$:*

1. $\Pr\left[|\boldsymbol{w} - \boldsymbol{c}| > \sigma\sqrt{m} \ : \ \boldsymbol{w} \xleftarrow{\$} \mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}\right] \leq \mathrm{negl}(n)$.
2. *A set of $O(m \log m)$ samples from $\mathcal{D}_{\Lambda_q^\perp(A),\sigma}$ contains a full rank set in $\mathbb{Z}^m$, except with negligible probability.*
3. *There is a PPT algorithm $\mathsf{SampleGaussian}(A, T_A, \sigma, \boldsymbol{c})$ that returns $\boldsymbol{x} \in \Lambda_q^\perp(A)$ drawn from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}$.*

4. *There is a PPT algorithm* $\mathsf{SamplePre}(A, T_A, \boldsymbol{u}, \sigma)$ *that returns* $\boldsymbol{x} \in \Lambda_q^{\perp}(A)$ *sampled from a distribution statistically close to* $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(A),\sigma}$*, whenever* $\Lambda_q^{\boldsymbol{u}}(A)$ *is not empty.*

Following [1, 6, 3, 4, 22] we will need the following algorithms to sample short vectors and random basis from specific lattices.

Algorithm TrapGen Ajtai [3], and later Alwen and Peikert [4], showed how to sample an essentially uniform matrix $A \in \mathbb{Z}_q^{n\times m}$ along with a basis S of $\Lambda_q^{\perp}(A)$ with low Gram-Schmidt norm.

Theorem 1. *([4, Theorem 3.2] with* $\delta = 1/3$*) Let* q, n, m *be positive integers with* $q \geq 2$ *and* $m \geq 6n \lg q$*. There is a probabilistic polynomial-time algorithm* $\mathsf{TrapGen}(q, n, m)$ *that outputs a pair* $(A \in \mathbb{Z}_q^{n\times m}, S \in \mathbb{Z}^{m\times m})$ *such that* A *is statistically close to uniform in* $\mathbb{Z}_q^{n\times m}$ *and* S *is a basis for* $\Lambda_q^{\perp}(A)$*, satisfying* $|\widetilde{S}| \leq O(\sqrt{n \log q})$ *and* $|S| \leq O(n \log q)$ *with overwhelming probability in* n*.*

Algorithm IdealTrapGen Stehlé et al. [22] showed an adaptation of Ajtai's trapdoor key generation algorithm for ideal lattices.

Theorem 2. *([22, Theorem 3.1]) Let* n, σ, q, k *be positive integers with* $q \equiv 3 \mod 8$*,* $k \geq \lceil \log q + 1 \rceil$*, let* n *be a power of 2 and let* $f(x) = x^n + 1$ *be a degree* n *polynomial in* $\mathbb{Z}[x]$*. Then, there is a probabilistic polynomial-time algorithm* $\mathsf{IdealTrapGen}(q, n, k, \sigma, f)$ *that outputs a pair* $(\hat{\boldsymbol{a}} \in \mathcal{R}^k, S \in \mathbb{Z}^{kn\times kn})$ *such that* $\hat{\boldsymbol{a}}$ *is statistically close to uniform in* $\mathcal{R}^k$ *and* S *is a basis for* $\Lambda_q^{\perp}(A)$*, for* $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}})$*, satisfying* $|S| = O(n \log q \sqrt{\omega(\log n)})$ *with overwhelming probability in* n*.*

Sampling Algorithms The following theorems give a few sample algorithms used in lattice-based schemes.

Theorem 3. *([1, Theorem 17], [6, Lemma 3.2]) Let* $q > 2, m > n$ *and* $\sigma > |T_A| \cdot \omega(\sqrt{\log(m + m_1)})$*. Then* $\mathsf{SampleLeft}(A, B, T_A, \boldsymbol{u}, \sigma)$ *outputs a vector* $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ *statistically close to* $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(F),\sigma}$*, with* $F = (A|B)$*.*

Theorem 4. *([1, Theorem 19]) Let* $q > 2, m > n$ *and* $\sigma > |T_B| \cdot |R| \cdot \omega(\sqrt{\log(k + m)})$*. Then* $\mathsf{SampleRight}(A, B, R, T_B, \boldsymbol{u}, \sigma)$ *outputs a vector* $\boldsymbol{e} \in \mathbb{Z}^{k+m}$ *distributed statistically close to* $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(F),\sigma}$*, with* $F = (A|AR + B)$*.*

Theorem 5. *[1] Let* $A \in \mathbb{Z}_q^{n\times m}$ *be a full rank matrix, let* S *be a short basis of* $\Lambda_q^{\perp}(A)$*, let* $B \in \mathbb{Z}_q^{n\times m_1}$ *be a matrix and let* σ *be a Gaussian parameter. For* q, m, n *be integers such that* $q > 2$ *and* $m > 2n \log q$ *and* $\sigma > \|S\| \cdot \omega(\sqrt{\log(m + m_1)})$*, then there is a probabilistic polynomial algorithm* $\mathsf{SampleBasisLeft}(A, B, S, \sigma)$ *that outputs a new basis* $T \in \mathbb{Z}^{n\times m+m_1}$ *for the lattice* $\Lambda_q^{\perp}(F)$*, with* $F = (A|B)$*.*

Theorem 6 ([1])**.** *Let* $A \in \mathbb{Z}_q^{n\times m}$ *and* $B \in \mathbb{Z}_q^{n\times m_1}$ *be a full rank matrices, let* S *be a short basis of* $\Lambda_q^{\perp}(B)$*, let* $R \in \{-1, 1\}^{m\times m1}$ *be a uniform random matrix and let* σ *be a Gaussian parameter. Let* q, m, n *be integers such that* $q > 2$ *and* $m > n$ *and let* $\sigma > \|S\| \cdot \sqrt{m} \cdot \omega(\sqrt{\log m})$*. Then there is a probabilistic polynomial algorithm* $\mathsf{SampleBasisRight}(A, B, R, S, \sigma)$ *that outputs a new basis* $T \in \mathbb{Z}^{n\times m+m_1}$ *for the lattice* $\Lambda_q^{\perp}(F)$*, with* $F = (A|AR + B)$*.*

The LWE Problem The Learning with Errors problem, or LWE, is the problem of determining a secret vector over $\mathbb{F}_q$ given a polynomial number of noisy inner products. The decision variant is to distinguish such samples from random. More formally, we define the (average-case) problem as follows:

Definition 1. *([19]) Let $n \geq 1$ and $q \geq 2$ be integers, and let χ be a probability distribution on $\mathbb{Z}_q$. For $\boldsymbol{r} \in \mathbb{Z}_q^n$, let $A_{\boldsymbol{r},\chi}$ be the probability distribution on $\mathbb{Z}_q^n \times \mathbb{Z}_q$ obtained by choosing a vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ uniformly at random, choosing $e \in \mathbb{Z}_q$ according to χ, and outputting $(\boldsymbol{a}, b \leftarrow \langle \boldsymbol{a}, \boldsymbol{r} \rangle + e)$.*

(a) The *search*-$\mathsf{LWE}_{q,n,\chi}$ problem is: for uniformly random $\boldsymbol{r} \in \mathbb{Z}_q^n$, given a $\mathrm{poly}(n)$ number of samples from $A_{\boldsymbol{r},\chi}$, output $\boldsymbol{r}$.

(b) The *decision*-$\mathsf{LWE}_{q,n,\chi}$ problem is: for uniformly random $\boldsymbol{r} \in \mathbb{Z}_q^n$, given a $\mathrm{poly}(n)$ number of samples that are either (all) from $A_{\boldsymbol{r},\chi}$ or (all) uniformly random in $\mathbb{Z}_q^n \times \mathbb{Z}_q$, output 0 if the former holds and 1 if the latter holds.

The hardness of the LWE problem is summarised in the following:

Definition 2. *For $\alpha \in (0,1)$ and an integer $q > 2$, let $\overline{\Psi}_\alpha$ denote the probability distribution over $\mathbb{Z}_q$ obtained by choosing $x \in \mathbb{R}$ according to the normal distribution with mean 0 and standard deviation $\alpha/\sqrt{2\pi}$ and outputting $\lfloor qx \rceil$.*

Theorem 7. *([19]) Let n, q be integers and $\alpha \in (0,1)$ such that $q = \mathrm{poly}(n)$ and $\alpha q > 2\sqrt{n}$. If there exists an efficient (possibly quantum) algorithm that solves decision-$\mathsf{LWE}_{q,n,\overline{\Psi}_\alpha}$, then there exists an efficient quantum algorithm that approximates SIVP and GSVP to within $\widetilde{O}(n/\alpha)$ in the worst case.*

The $\mathsf{RingLWE}$ Problem is the same as described above, but with in the set $\mathcal{R}$ with $\boldsymbol{b} \leftarrow \boldsymbol{a} \otimes \boldsymbol{r} + \boldsymbol{e}$.

Theorem 8 ([12])**.** *Let n, q be integers and $\alpha > 0$ such that $q \geq 2$, $q = 1 \mod m$ and q be a $\mathrm{poly}(n)$-bounded prime such that $\alpha q \geq \omega(\sqrt{\log n})$. If there exists an efficient (possibly quantum) algorithm that solves decision-$\mathsf{RingLWE}_{q,n,\Upsilon_\alpha}$, then there exists an efficient quantum algorithm that solves γ-SIVP and γ-SVP for $\gamma = \widetilde{O}(n/\alpha)$ in the worst case.*

The following lemmas are use to bound the norm of vectors and matrices and will be used to show correctness of decryption.

Lemma 5. *([1, Lemma 12]) Let $\boldsymbol{e}$ be some vector in $\mathbb{Z}^m$ and let $\boldsymbol{v} \leftarrow \overline{\Psi}_\alpha^m$. Then, the quantity $|\langle \boldsymbol{e}, \boldsymbol{v} \rangle|$, when treated as an integer in $(-q/2, q/2]$, satisfies $|\langle \boldsymbol{e}, \boldsymbol{v} \rangle| \leq |\boldsymbol{e}| \cdot \big(q\alpha \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2\big)$ with overwhelming probability (in m).*

Lemma 6. *Let $\boldsymbol{r}$ be a vector of length n in $\{-1, 1\}^n$; then, $|\mathsf{rot}(\boldsymbol{r})| \leq O(n\sqrt{n})$ is a bound for the norm of $\mathsf{rot}(\boldsymbol{r})$.*

Lemma 7. *Let $R = \mathsf{Exp}(\hat{\boldsymbol{r}})$ be a $kn \times kn$ matrix, for $\hat{\boldsymbol{r}} \in \mathcal{R}^k$; then, $|R| \leq \sqrt{k} \max |R_i|$ is a bound for the norm of R, with $R_i = \mathsf{rot}_f(\boldsymbol{r}_i)$.*

3 Our Identity-Based Encryption Scheme

As described in Section 2.1, an Identity Based Encryption Scheme consists of the following algorithms: SetUp, KeyGen, Enc and Dec. In this section we describe each algorithm for our construction, based on the IBE scheme presented by Agrawal et al. [1].

The SetUp algorithm creates an ideal lattice based on a function $f(x) = x^n + 1$, with n a power of 2. It chooses vectors at random rather than matrices, as the original scheme does, which results in smaller public keys. The KeyGen algorithm generates the secret key by encoding the identity id into a matrix using the $\mathsf{rot}_f()$ function and concatenating it to the lattice basis. The secret key is the vector created by the sample algorithm, i.e., $\boldsymbol{e} \in \Lambda_q^{\boldsymbol{u}}(A_{id})$.

The Enc algorithm uses the message, the identity id and the vectors in the public-key to create an integer c' and a vector $\boldsymbol{c}$ that will compose the ciphertext ct for one bit. Finally, the Dec algorithm can recover the message from the ciphertext, only if the identity used during the key generation is exactly the same as the one used in the encryption.

3.1 Our Construction

Let $n = 2^\alpha$ be the security parameter, σ be the Gaussian parameter and $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$, with $f(x) = \langle x^n + 1 \rangle$. Notice that $[\hat{\boldsymbol{x}}|\boldsymbol{x}_0 R_0| \dots |\boldsymbol{x}_{k-1} R_{k-1}] = [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R]$, for $R = \mathsf{Exp}(\hat{\boldsymbol{r}})$ and $R_i = \mathsf{rot}_f(\boldsymbol{r}_i)$.

SetUp(1^n)

1. run the IdealTrapGen(n, k, q, f, σ) algorithm to select uniformly a vector $\hat{\boldsymbol{a}} \in \mathcal{R}^k$, with a short basis $T \in \mathbb{Z}^{kn \times kn}$ for $\Lambda_q^{\perp}(A)$, such that $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}})$;
2. choose uniformly random vectors $\hat{\boldsymbol{a}}', \hat{\boldsymbol{b}} \in \mathcal{R}_q^k$ and $\boldsymbol{u} \in \mathcal{R}_q$;
3. output $mpk = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}', \hat{\boldsymbol{b}}, \boldsymbol{u})$ and $msk = T$.

KeyGen($mpk, msk, id = \hat{d}$)

1. sample a vector for lattice $\Lambda(A_{id})_q^{\boldsymbol{u}}$, with $A_{id} = [A|C]$, where $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}})$ and $C = [C_0| \dots |C_{k-1}]$, for $C_i = \mathsf{rot}_f(\boldsymbol{a}'_i) + \mathsf{rot}_f(\boldsymbol{b}_i \otimes \boldsymbol{d}_i)$ by invoking $\hat{\boldsymbol{e}} \leftarrow \mathsf{SampleLeft}(A, C, T, \boldsymbol{u}, \sigma)$;
2. output $sk = \hat{\boldsymbol{e}}$.

Enc($mpk, msg, id = \hat{d}$)

1. choose a uniformly random vector $\boldsymbol{s} \in \mathcal{R}_q$ and a random ring $\hat{\boldsymbol{r}} \in \{-1, 1\}^{kn}$;
2. choose a noise vector $\hat{\boldsymbol{x}} \in \overline{\Psi}_{\alpha_t}^{kn}$ and a noise term $x \in \overline{\Psi}_{\alpha_t}$;
3. compute $\hat{\boldsymbol{c}} = A_{id}^{\top} \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R]$, for $R = \mathsf{Exp}(\hat{\boldsymbol{r}})$ and $A_{id} = [A|C]$;
4. compute $c' = \boldsymbol{u}^{\top} \boldsymbol{s} + x + msg \cdot \lfloor q/2 \rceil$;
5. output $ct = (\hat{\boldsymbol{c}}, c')$.

Dec(mpk, sk, ct)

1. compute $z = c' - \hat{\boldsymbol{e}}^\top \hat{\boldsymbol{c}}$; for $z \in (-q/2, q/2]$;
2. output 0 if $|z| < q/4$ and 1 otherwise.

3.2 Correctness

If the exactly same identity is used during key generation and encryption, we have:

$$\begin{aligned} z &= c' - \hat{\boldsymbol{e}}^\top \hat{\boldsymbol{c}} \\ &= \boldsymbol{u}^\top \boldsymbol{s} + x + msg \cdot \lfloor q/2 \rceil - \hat{\boldsymbol{e}}^\top A_{id}^\top \boldsymbol{s} - \hat{\boldsymbol{e}}^\top [\hat{\boldsymbol{x}} | \hat{\boldsymbol{x}} R] \\ &= \boldsymbol{u}^\top \boldsymbol{s} + x + msg \cdot \lfloor q/2 \rceil - \boldsymbol{u}^\top \boldsymbol{s} - \hat{\boldsymbol{e}}^\top [\hat{\boldsymbol{x}} | \hat{\boldsymbol{x}} R] \\ &= msg \cdot \lfloor q/2 \rceil + \underbrace{x - \hat{\boldsymbol{e}}^\top [\hat{\boldsymbol{x}} | \hat{\boldsymbol{x}} R]}_{\text{error term}} \end{aligned}$$

Note that for the correct decryption the error term must be less than $q/4$.

3.3 Security

In this section we prove the following theorem.

Theorem 9. *If the decision-*RingLWE *problem is infeasible, then the IBE scheme described on Section 3.1 is IND-sID-CPA.*

Following [1], we define additional algorithms. These will not be used in the real scheme, but we need them in our proofs.

Sim.SetUp(1^n, $id^\star$): The algorithm chooses random $\hat{\boldsymbol{a}} \in \mathcal{R}_q^k$, $\boldsymbol{u} \in \mathcal{R}_q$ and $\hat{\boldsymbol{r}}^\star \in \{-1, 1\}^{kn \times kn}$; it uses IdealTrapGen to generate $\hat{\boldsymbol{b}}^\star \in \mathcal{R}_q^k$ with a basis $T \in \mathbb{Z}^{kn \times kn}$ for $\Lambda_q^\perp(B^\star)$, where $B^\star = \mathsf{Rot}_f(\hat{\boldsymbol{b}}^\star)$ and defines $\boldsymbol{a}_i' \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i^\star$. Then, it outputs $mpk = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}', \boldsymbol{u})$ and $msk = (\hat{\boldsymbol{r}}^\star, \hat{\boldsymbol{b}}^\star, T)$.
Sim.KeyGen(mpk, msk, id): Secret keys are now created by using the trapdoor T, sampled by Sim.SetUp, and the SampleRight algorithm. It outputs $sk = \hat{\boldsymbol{e}} \in \Lambda_q^{\boldsymbol{u}}(A|AR^\star + B^\star D)$, where $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}}^\star)$, $B^\star = \mathsf{Rot}_f(\hat{\boldsymbol{b}}^\star)$, $R^\star = \mathsf{Exp}(\hat{\boldsymbol{r}}^\star)$ and $D = \mathsf{Exp}(\hat{\boldsymbol{d}} - \hat{\boldsymbol{d}}^\star)$, for $\hat{\boldsymbol{d}} - \hat{\boldsymbol{d}}^\star \neq \boldsymbol{0}$, by invoking $\hat{\boldsymbol{e}} \leftarrow \mathsf{SampleRight}(A, B^\star D, R^\star, T, \boldsymbol{u}, \sigma)$. By construction, each $D_i = \mathsf{rot}_f(\boldsymbol{d}_i)$ and $D_i^\star = \mathsf{rot}_f(\boldsymbol{d}_i^\star)$ are full-rank and non-singular matrices; therefore, $\mathsf{Exp}(D - D^\star)$ is also a full-rank non-singular matrix. Note that $det(D) = \prod det(D_i - D_i^\star)$ and, since each $(D_i - D_i^\star)$ is a full-rank non singular matrix (see Section 5 of [1]), then D is also full-rank non-singular.
Sim.Enc(mpk, msg, id): The algorithm differs from Enc in the sense that it uses rings $\hat{\boldsymbol{r}}^\star$ and $\hat{\boldsymbol{b}}^\star$ instead of tings $\hat{\boldsymbol{r}}$ and $\hat{\boldsymbol{b}}$.

For a probabilistic polynomial-time adversary $\mathcal{A}$, our proof of security will consist of the following sequence of six games between $\mathcal{A}$ and $\mathcal{C}$. The six games are defined as follows:

Game 0 $\mathcal{C}$ runs the SetUp algorithm, answers $\mathcal{A}$'s secret key queries using the KeyGen algorithm, and generates the challenge ciphertext using Enc with vector $id^\star = id^0$ and message msg_0.

Game 1 In this game, $\mathcal{C}$ uses the simulation algorithms. Specifically, $\mathcal{C}$ runs the $\mathsf{Sim.SetUp}$ algorithm with id^0, answers $\mathcal{A}$'s secret key queries using the $\mathsf{Sim.KeyGen}$ algorithm, and generates the challenge ciphertext using $\mathsf{Sim.Enc}$ with $id^\star = id^0$ and message msg_0.

Game 2 This is the same as Game 1, except that the challenge ciphertext is randomly chosen from the ciphertext space.

Game 3 Same as Game 2, except that $\mathcal{C}$ runs the $\mathsf{Sim.SetUp}$ algorithm with $id^\star = id^1$.

Game 4 Same as Game 3, except that $\mathcal{C}$ generates the challenge ciphertext using $\mathsf{Sim.Enc}$ with $id^\star = id^1$ and message msg_1.

Game 5 $\mathcal{C}$ runs the SetUp algorithm, answers $\mathcal{A}$'s secret key queries using the KeyGen algorithm, and generates the challenge ciphertext using Enc with $id^\star = id^1$ and message msg_1.

We have that for $i = \{0, \ldots, 4\}$, Game i is indistinguishable from Game $i+1$ under the appropriate assumptions.

3.3.1 Indistinguishability of Game 0 and Game 1 (or Game 4 and Game 5)

Lemma 8. *The view of adversary $\mathcal{A}$ in Game 0 (resp. Game 4) is statistically close to the view of $\mathcal{A}$ in Game 1 (resp. Game 5).*

Proof.

Setup In Game 0, the ring $\hat{\boldsymbol{a}}$ is generated by $\mathsf{IdealTrapGen}$ and ring $\hat{\boldsymbol{a}}'$ is uniformly random in $\mathcal{R}_q^k$. On the other hand, in Game 1, $\hat{\boldsymbol{a}}$ is chosen uniformly at random and $\hat{\boldsymbol{a}}'$ is the concatenation of $\boldsymbol{a}'_i \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i^\star$. In both games vector $\boldsymbol{u}$ is random. Notice that, by Theorem 2, vectors $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}^\star$, output by $\mathsf{IdealTrapGen}$, are statistically indistinguishable from a uniformly random vector.

Secret keys The secret key in Game 0 for identity id^0 is the ring $\hat{\boldsymbol{e}} \in \Lambda_q^{\boldsymbol{u}}(A_{id})$, where $A_{id} = [A|C]$ is sampled using the $\mathsf{SampleLeft}$ algorithm. In Game 1, the secret key for identity id^0 is the ring $\hat{\boldsymbol{e}} \in \Lambda_q^{\boldsymbol{u}}(A|AR^\star + B^\star D)$, sampled with the $\mathsf{SampleRight}$ algorithm. Thus, the secret keys have the same distribution in both games.

Challenge Ciphertext In both games the challenge ciphertext component c' and $\hat{\boldsymbol{c}}$ are computed the same way:

$$\begin{aligned}\hat{\boldsymbol{c}} &= A_{id}^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R] \\ &= [A|C_0|\cdots|C_{k-1}]^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\boldsymbol{x}_0 R_0|\cdots|\boldsymbol{x}_{k-1}R_{k-1}] \in \mathcal{R}_q^{2k}\end{aligned}.$$

But, in Game 0, the matrices C_i is computed as follows:

$$\begin{aligned}C_i &= \mathsf{rot}_f(\boldsymbol{a}'_i) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}'_i + \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i).\end{aligned}.$$

On the other hand, in Game 1, we have:

$$\begin{aligned} C_i &= \mathsf{rot}_f(\boldsymbol{a}'_i) + \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i) + \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i) - \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i) + \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i). \end{aligned} .$$

Let us now analyse the joint distribution of the public parameters and the challenge ciphertext in Game 0 and Game 1. We will show that the distributions of $(\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}', \hat{\boldsymbol{c}})$ in Game 0 and in Game 1 are statistically indistinguishable.

First notice that, by Lemma 3, we have that the following two distributions are statistically indistinguishable for every fixed $\hat{\boldsymbol{b}}^\star$ and $\hat{\boldsymbol{d}}^\star$:

$$\begin{gathered} (\boldsymbol{a}_i, \boldsymbol{a}'_i, [\boldsymbol{x}_i | \boldsymbol{x}_i R^\star_i]) \approx_s \\ (\boldsymbol{a}_i, \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i, [\boldsymbol{x}_i | \boldsymbol{x}_i R^\star_i]) \ . \end{gathered}$$

Since $[\mathsf{rot}_f(\boldsymbol{a}_i) | \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i)]^\top \boldsymbol{s}$ is statistically close to $[\mathsf{rot}_f(\boldsymbol{a}_i) | \mathsf{rot}_f(\boldsymbol{a}'_i)]^\top \boldsymbol{s}$, it is possible to add each term to one side of the equation:

$$\begin{gathered} (\boldsymbol{a}_i, \boldsymbol{a}'_i, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i | (\mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i))^\top \boldsymbol{s} + \boldsymbol{x}_i R^\star_i]) \approx_s \\ (\boldsymbol{a}_i, \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i | \mathsf{rot}_f(\boldsymbol{a}'_i)^\top \boldsymbol{s} + \boldsymbol{x}_i R^\star_i]) \ . \end{gathered}$$

Finally, we add $\mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i)^\top \boldsymbol{s}$ to both sides:

$$\begin{gathered} (\boldsymbol{a}_i, \boldsymbol{a}'_i, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i | (\mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i))^\top \boldsymbol{s} + \boldsymbol{x}_i R^\star_i]) \approx_s \\ (\boldsymbol{a}_i, \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i | \mathsf{rot}_f(\boldsymbol{a}'_i + \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i)^\top \boldsymbol{s} + \boldsymbol{x}_i R^\star_i]) \ . \end{gathered}$$

To conclude, observe that if we concatenate all the parts, we have that the distribution on the left hand side is that of the public parameters and the challenge ciphertext in Game 0, while that on the right hand side is the distribution in Game 1.

□

3.3.2 Indistinguishability of Game 1 and Game 2 (or Game 3 and Game 4)

Lemma 9. *The view of adversary $\mathcal{A}$ in Game 1 (resp. Game 3) is computationally indistinguishable from the view of $\mathcal{A}$ in Game 2 (resp. Game 4) under decision-*RingLWE.

Proof.

Suppose $\mathcal{A}$ can distinguish between Game 1 and Game 2 with non-negligible advantage. Then, it is possible to use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ to solve decision-RingLWE.

Init $\mathcal{B}$ is given k RingLWE challenge pairs $(\boldsymbol{a}_j, \boldsymbol{y}_j) \in \mathcal{R}_q \times \mathcal{R}_q$, where either $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$ for a random $\boldsymbol{s} \in \mathcal{R}_q$ and a noise term $\boldsymbol{x}_j \leftarrow \Psi^n_\alpha$, or $\boldsymbol{y}_j$ is uniformly random in $\mathcal{R}_q$. And one LWE challenge pair $(\boldsymbol{a}_0, y_0) \in \mathbb{Z}^n_q \times \mathbb{Z}_q$, where either $y_0 = \langle \boldsymbol{a}_0, \boldsymbol{s} \rangle + x_0$ for a noise term $x_0 \leftarrow \Psi_\alpha$, or y_0 is uniformly random in $\mathbb{Z}_q$.

SetUp The public parameters are constructed using the vectors of the pairs $(\boldsymbol{a}_j, \boldsymbol{y}_j)$. The i-th polynomial of ring $\hat{\boldsymbol{a}}$ will be the vector $\boldsymbol{a}_i$, for $1 \leq i \leq k$ and vector $\boldsymbol{u}$ will be $\boldsymbol{a}_0$. The ring $\hat{\boldsymbol{a}}'$ is still calculated as in Sim.SetUp, i.e., $\boldsymbol{a}'_i \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i$.

Secret keys All private-key extraction queries are answered using $\mathsf{Sim.KeyGen}$.

Challenge Ciphertext The ciphertext $CT = (\hat{\boldsymbol{c}}^\star, c'^\star)$ is constructed based on the terms in the LWE challenge pairs $(\boldsymbol{a}_j, \boldsymbol{y}_j)$, with $\hat{\boldsymbol{c}}^\star = [(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)|R^{\star\top}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)]$ and $c'^\star = y_0 + M\lfloor q/2 \rceil$. If we have $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$ on the $\mathsf{RingLWE}$ challenge and $y_0 = \langle \boldsymbol{a}_0, \boldsymbol{s} \rangle + x_0$ on the LWE challenge, then the ciphertext is distributed exactly as in Game 1, and if $\boldsymbol{y}_j$ is uniformly random in $\mathcal{R}_q$ and y_0 is uniformly random in $\mathbb{Z}_q$, then the ciphertext is distributed exactly as in Game 2.

If $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$, then $(y_1, \ldots, y_m) = (\boldsymbol{a}_1 \otimes \boldsymbol{s} + \boldsymbol{x}_1, \ldots, \boldsymbol{a}_k \otimes \boldsymbol{s} + \boldsymbol{x}_m) = \mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}$. Therefore, for Game 1 we have

$$\begin{aligned} C_i &= \mathsf{rot}_f(\boldsymbol{a}'_i) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i^\star) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star) - \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i^\star) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i)\mathsf{rot}_f(\boldsymbol{r}_i^\star). \end{aligned}$$

and

$$\begin{aligned} \hat{\boldsymbol{c}} &= A_{id}^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R^\star] \\ &= [A|C_0|\cdots|C_{k-1}]^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R^\star] \\ &= [\mathsf{Rot}_f(\hat{\boldsymbol{a}})|\mathsf{rot}_f(\boldsymbol{a}_0)\mathsf{rot}_f(\boldsymbol{r}_0^\star)|\cdots|\mathsf{rot}_f(\boldsymbol{a}_{k-1})\mathsf{rot}_f(\boldsymbol{r}_{k-1}^\star)]^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R^\star] \\ &= [\mathsf{Rot}_f(\hat{\boldsymbol{a}})|R^{\star\top}\mathsf{Rot}_f(\hat{\boldsymbol{a}})]^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R^\star] \\ &= [\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}|R^{\star\top}(\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}})] \\ &= [(y_1, \ldots, y_m)|R^{\star\top}(y_1, \ldots, y_m)]\ . \end{aligned}$$

If y_j is uniformly random in $\mathbb{Z}_q$ then the ciphertext is uniformly random, as the ciphertext generated by Game 2.

Guess $\mathcal{A}$ must guess whether it is interacting with Game 1 or Game 2. The answer to this guess is also the answer to the $\mathsf{RingLWE}$ and LWE challenges, because, as we showed, if $\boldsymbol{y}_j$ is uniformly random in $\mathcal{R}_q$ and y_0 is uniformly random in $\mathbb{Z}_q$, then $\mathcal{A}$'s view is the same as in Game 2 and if $y_0 = \langle \boldsymbol{a}_0, \boldsymbol{s} \rangle + x_0$ and $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$, then $\mathcal{A}$'s view is the same as in Game 1.

□

3.3.3 Indistinguishability of Game 2 and Game 3

Lemma 10. *The view of adversary $\mathcal{A}$ in Game 2 is statistically indistinguishable from the view of $\mathcal{A}$ in Game 3*

Proof.

SetUp The public parameters are generated in the same way in both games. Vectors $\hat{\boldsymbol{a}}$ and $\boldsymbol{u}$ are random and $\boldsymbol{a}'_i \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_i^\star$, with $\hat{\boldsymbol{d}}^\star = \hat{\boldsymbol{d}}^0$ for Game 2 and $\hat{\boldsymbol{d}}^\star = \hat{\boldsymbol{d}}^1$ for Game 3.

Secret keys All private-key extraction queries are answered using $\mathsf{Sim.KeyGen}$. The only difference is, again, that for Game 2, $id^\star = id^0$ and, for Game 3, $id^\star = id^1$.

Challenge Ciphertext The challenge ciphertext in both games is randomly chosen.

All public parameters are randomly generated in both games, except for ring $\hat{\boldsymbol{a}}'$. Therefore, the indistinguishability of Game 2 and Game 3 only depends on the indistinguishability of $\hat{\boldsymbol{a}}'$. From Lemma 3 we can prove that $\hat{\boldsymbol{a}}'$ for each game is statistically close to a uniformly random ring, because

$$\boldsymbol{a}'_i \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_i - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_i.$$

□

4 Our Hierarchical Identity-Based Encryption Scheme

As described in Section 2.1, an Hierachical Identity Based Encryption Scheme consists of the following algorithms: SetUp, $\mathsf{KeyDerive}$, Enc and Dec. In this section we describe each algorithm for our construction, based on the HIBE scheme presented by Agrawal et al. [1].

The SetUp algorithm creates an ideal lattice based on a function $f(x) = x^n + 1$, with n a power of 2. It chooses vectors at random rather than matrices, as the original scheme does, which results in smaller public keys. The $\mathsf{KeyDerive}$ algorithm generates the secret key by encoding each identity id_i into a matrix using the $\mathsf{rot}_f()$ function and concatenating it to the lattice basis. Now, the secret key is a short basis for the lattice generate by this encoding, using the algorithm $\mathsf{SampleBasisLeft}$. Note that $sk_0 = msk$,.

The Enc algorithm uses the message, the identity id and the vectors in the public-key to create an integer c' and a vector $\boldsymbol{c}$ that will compose the ciphertext ct for one bit. Finally, the Dec algorithm can recover the message from the ciphertext, only if all the identities used during the key generation are exactly the same as the ones used in the encryption.

4.1 Our Construction

Let $n = 2^\alpha$ be the security parameter, μ be the hierarchical parameter, σ_j be the Gaussian parameters and $\mathcal{R} = \mathbb{Z}_q[x]/f(x)$, with $f(x) = \langle x^n{+}1\rangle$. Notice that $[\hat{\boldsymbol{x}}|\boldsymbol{x}_0 R_{0,j}|\ldots|\boldsymbol{x}_{k-1}R_{k-1,j}] = [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_j]$, for $R_j = \mathsf{Exp}(\hat{\boldsymbol{r}}_j)$ and $R_{i,j} = \mathsf{rot}_f(\boldsymbol{r}_{i,j})$.

$\mathsf{SetUp}(1^n, 1^\mu)$

1. run the $\mathsf{IdealTrapGen}(n, k, q, f, \sigma)$ algorithm to select uniformly a vector $\hat{\boldsymbol{a}} \in \mathcal{R}^k$, with a short basis $T \in \mathbb{Z}^{kn\times kn}$ for $\Lambda^\perp_q(A)$, such that $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}})$;
2. choose uniformly random vectors $\hat{\boldsymbol{a}}'_j, \hat{\boldsymbol{b}} \in \mathcal{R}^k_q$ and $\boldsymbol{u} \in \mathcal{R}_q$, for $j \in [1, h]$;
3. output $mpk = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}'_j, \hat{\boldsymbol{b}}, \boldsymbol{u})$ and $msk = T$.

$\mathsf{KeyDerive}(mpk, sk_{t-1}, id_1 = \hat{d}_1, \ldots, id_t = \hat{d}_t)$

1. sample a short basis for lattice $\Lambda(A_{id}|C_t)_q^{\perp}$, with $A_{id} = [A|C_1|\cdots|C_{t-1}]$, where $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}})$ and $C_j = [C_{0,j}|\ldots|C_{k-1,j}]$, for $C_{i,j} = \mathsf{rot}_f(\boldsymbol{a}'_{i,j}) + \mathsf{rot}_f(\boldsymbol{b}_i \otimes \boldsymbol{d}_{i,j})$ by invoking $S_t \leftarrow \mathsf{SampleBasisLeft}(A_{id}, C_t, T, S_{t-1}, \sigma_t)$;
2. output $sk = S_t$.

$\mathsf{Enc}(mpk, msg, id_1 = \hat{d}_1, \ldots, id_t = \hat{d}_t)$

1. choose a uniformly random vector $\boldsymbol{s} \in \mathcal{R}_q$ and random rings $\hat{\boldsymbol{r}}_j \in \{-1, 1\}^{kn}$;
2. choose a noise vector $\hat{\boldsymbol{x}} \in \overline{\Psi}_{\alpha_t}^{kn}$ and a noise term $x \in \overline{\Psi}_{\alpha_t}$;
3. compute $\hat{\boldsymbol{c}} = A_{id}^{\top}\boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_1|\cdots|\hat{\boldsymbol{x}}R_t]$, for $R_j = \mathsf{Exp}(\hat{\boldsymbol{r}}_j)$ and $A_{id} = [A|C_1|\cdots|C_t]$;
4. compute $c' = \boldsymbol{u}^{\top}\boldsymbol{s} + x + msg \cdot \lfloor q/2 \rceil$;
5. output $ct = (\hat{\boldsymbol{c}}, c')$.

$\mathsf{Dec}(mpk, sk_t, ct)$

1. sample a vector for lattice $\Lambda(A_{id})_q^{\boldsymbol{u}}$, with $A_{id} = [A|C_1|\cdots|C_t]$, using the short basis on the secret key, by invoking algorithm $\hat{\boldsymbol{e}} \leftarrow \mathsf{SamplePre}(A_{id}, S_t, \boldsymbol{u}, \sigma)$, with $\sigma = \sigma_t\sqrt{kn(t+1)}\omega(\sqrt{\log(tkn)})$;
2. compute $z = c' - \hat{\boldsymbol{e}}^{\top}\hat{\boldsymbol{c}}$; for $z \in (-q/2, q/2]$;
3. output 0 if $|z| < q/4$ and 1 otherwise.

4.2 Correctness

If the exactly same identity is used during key generation and encryption, we have:

$$\begin{aligned} z &= c' - \hat{\boldsymbol{e}}^{\top}\hat{\boldsymbol{c}} \\ &= \boldsymbol{u}^{\top}\boldsymbol{s} + x + msg \cdot \lfloor q/2 \rceil - \hat{\boldsymbol{e}}^{\top}A_{id}^{\top}\boldsymbol{s} - \hat{\boldsymbol{e}}^{\top}[\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_1|\cdots|\hat{\boldsymbol{x}}R_t] \\ &= \boldsymbol{u}^{\top}\boldsymbol{s} + x + msg \cdot \lfloor q/2 \rceil - \boldsymbol{u}^{\top}\boldsymbol{s} - \hat{\boldsymbol{e}}^{\top}[\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_1|\cdots|\hat{\boldsymbol{x}}R_t] \\ &= msg \cdot \lfloor q/2 \rceil + \underbrace{x - \hat{\boldsymbol{e}}^{\top}[\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_1|\cdots|\hat{\boldsymbol{x}}R_t]}_{\text{error term}} \end{aligned}$$

Note that for the correct decryption the error term must be less than $q/4$.

4.3 Security

In this section we prove the following theorem.

Theorem 10. *If the decision-*RingLWE *problem is infeasible, then the HIBE scheme described on Section 4.1 is IND-sID-CPA.*

Following [1], we define additional algorithms. These will not be used in the real scheme, but we need them in our proofs.

$\mathsf{Sim.SetUp}(1^n, id_1^{\star}, \cdots, id_h^{\star})$: The algorithm chooses random $\hat{\boldsymbol{a}} \in \mathcal{R}_q^k$, $\boldsymbol{u} \in \mathcal{R}_q$ and $\hat{\boldsymbol{r}}_{\boldsymbol{j}}^{\star} \in \{-1, 1\}^{kn \times kn}$; it uses IdealTrapGen to generate $\hat{\boldsymbol{b}}^{\star} \in \mathcal{R}_q^k$ with a basis $T \in \mathbb{Z}^{kn \times kn}$ for $\Lambda_q^{\perp}(B^{\star})$,

where $B^\star = \mathsf{Rot}_f(\hat{\boldsymbol{b}}^\star)$ and defines $\boldsymbol{a}'_{i,j} \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j} - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}$. Then, it outputs $mpk = (\hat{\boldsymbol{a}}, \hat{\boldsymbol{a}}'_j, \boldsymbol{u})$ and $msk = (\hat{\boldsymbol{r}}^\star_{\boldsymbol{j}}, \hat{\boldsymbol{b}}^\star, T)$.

Sim.KeyDerive($mpk, msk, id_1, \cdots, id_t$): Secret keys are now created by using the trapdoor T, sampled by Sim.SetUp, and the SampleBasisRight algorithm. It outputs $sk = S_t$,a short basis for lattice $\Lambda^{\boldsymbol{u}}_q(A|AR^\star_1 + B^\star D_1|\cdots|AR^\star_t + B^\star D_t)$, where $A = \mathsf{Rot}_f(\hat{\boldsymbol{a}}^\star)$, $B^\star = \mathsf{Rot}_f(\hat{\boldsymbol{b}}^\star)$, $R^\star_j = \mathsf{Exp}(\hat{\boldsymbol{r}}^\star_{\boldsymbol{j}})$ and $D_j = \mathsf{Exp}(\hat{\boldsymbol{d}}_j - \hat{\boldsymbol{d}}^\star_j)$, for $\hat{\boldsymbol{d}}_j - \hat{\boldsymbol{d}}^\star_j \neq \boldsymbol{0}$, by invoking $S_t \leftarrow \mathsf{SampleBasisRight}(A, B^\star_{id}, R^\star, T, \boldsymbol{u}, \sigma)$, for $R^\star = [R^\star_1|\cdots|R^\star_t]$ and $B^\star_{id} = [B^\star D_1|\cdots|B^\star D_t]$. By construction, each $D_{i,j} = \mathsf{rot}_f(\boldsymbol{d}_{i,j})$ and $D^\star_{i,j} = \mathsf{rot}_f(\boldsymbol{d}^\star_{i,j})$ are full-rank and non-singular matrices; therefore, $\mathsf{Exp}(D_j - D^\star_j)$ is also a full-rank non-singular matrix. Note that $det(D_j) = \prod det(D_{i,j} - D^\star_{i,j})$ and, since each $(D_{i,j} - D^\star_{i,j})$ is a full-rank non singular matrix (see Section 5 of [1]), then each D_j is also full-rank non-singular.

Sim.Enc(mpk, msg, id): The algorithm differs from Enc in the sense that it uses vectors $\hat{\boldsymbol{r}}^\star_j$ and $\hat{\boldsymbol{b}}^\star$ instead of vectors $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{b}}$.

For a probabilistic polynomial-time adversary $\mathcal{A}$, our proof of security will consist of the following sequence of six games between $\mathcal{A}$ and $\mathcal{C}$. The six games are defined as follows:

Game 0 $\mathcal{C}$ runs the SetUp algorithm, answers $\mathcal{A}$'s secret key queries using the KeyGen algorithm, and generates the challenge ciphertext using Enc with vector $id^\star_j = id^0_j$ and message msg_0.

Game 1 In this game, $\mathcal{C}$ uses the simulation algorithms. Specifically, $\mathcal{C}$ runs the Sim.SetUp algorithm with id^0_j, answers $\mathcal{A}$'s secret key queries using the Sim.KeyGen algorithm, and generates the challenge ciphertext using Sim.Enc with vector $id^\star_j = id^0_j$ and message msg_0.

Game 2 This is the same as Game 1, except that the challenge ciphertext is randomly chosen from the ciphertext space.

Game 3 Same as Game 2, except that $\mathcal{C}$ runs the Sim.SetUp algorithm with $id^\star_j = id^1_j$.

Game 4 Same as Game 3, except that $\mathcal{C}$ generates the challenge ciphertext using Sim.Enc with $id^\star_j = id^1_j$ and message msg_1.

Game 5 $\mathcal{C}$ runs the SetUp algorithm, answers $\mathcal{A}$'s secret key queries using the KeyGen algorithm, and generates the challenge ciphertext using Enc with $id^\star_j = id^1_j$ and message msg_1.

We have that for $i = \{0, \ldots, 4\}$, Game i is indistinguishable from Game $i + 1$ under the appropriate assumptions.

4.3.1 Indistinguishability of Game 0 and Game 1 (or Game 4 and Game 5)

Lemma 11. *The view of adversary $\mathcal{A}$ in Game 0 (resp. Game 4) is statistically close to the view of $\mathcal{A}$ in Game 1 (resp. Game 5).*

Proof.

Setup In Game 0, the ring $\hat{\boldsymbol{a}}$ is generated by IdealTrapGen and rings $\hat{\boldsymbol{a}}'_j$ are uniformly random in $\mathcal{R}^k_q$. On the other hand, in Game 1, $\hat{\boldsymbol{a}}$ is chosen uniformly at random and $\hat{\boldsymbol{a}}'_j$ are the concatenation of $\boldsymbol{a}'_{i,j} \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j} - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}$. In both games vector $\boldsymbol{u}$ is random.

Notice that, by Theorem 2, vectors $\hat{\boldsymbol{a}}$ and $\hat{\boldsymbol{b}}^\star$, output by $\mathsf{IdealTrapGen}$, are statistically indistinguishable from a uniformly random vector.

Secret keys The secret key in Game 0 for identities id_j^0 is the matrix S_t, a short basis for lattice $\Lambda_q^{\boldsymbol{u}}(A_{id})$, where $A_{id} = [A|C_1|\cdots|C_t]$, that is sampled using the $\mathsf{SampleBasisLeft}$ algorithm. In Game 1, the secret key for identities id_j^0 is the matrix S_t, a short basis for lattice $\Lambda_q^{\boldsymbol{u}}(A|AR_1^\star + B^\star D_1|\cdots|AR_t^\star + B^\star D_t)$, sampled with the $\mathsf{SampleBasisRight}$ algorithm. Thus, the secret keys have the same distribution in both games.

Challenge Ciphertext In both games the challenge ciphertext component c' and $\hat{\boldsymbol{c}}$ are computed the same way:

$$\begin{aligned}\hat{\boldsymbol{c}} &= A_{id}^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R] \\ &= [A|C_{0,1}|\cdots|C_{k-1,1}|\cdots|C_{0,t}|\cdots|C_{k-1,t}]^\top \boldsymbol{s} + \\ &\quad [\hat{\boldsymbol{x}}|\boldsymbol{x}_0 R_{0,1}|\cdots|\boldsymbol{x}_{k-1}R_{k-1,1}|\cdots|\boldsymbol{x}_0 R_{0,t}|\cdots|\boldsymbol{x}_{k-1}R_{k-1,t}] \in \mathcal{R}_q^{2tk}\end{aligned} .$$

But, in Game 0, the matrices $C_{i,j}$ is computed as follows:

$$\begin{aligned}C_{i,j} &= \mathsf{rot}_f(\boldsymbol{a}'_{i,j}) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}) \\ &= \mathsf{rot}_f(\boldsymbol{a}'_{i,j} + \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}).\end{aligned} .$$

On the other hand, in Game 1, we have:

$$\begin{aligned}C_{i,j} &= \mathsf{rot}_f(\boldsymbol{a}'_{i,j}) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_i^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star) - \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star) + \mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}) \\ &= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star).\end{aligned} .$$

Let us now analyse the joint distribution of the public parameters and the challenge ciphertext in Game 0 and Game 1. We will show that the distributions of $(\hat{\boldsymbol{a}}, \{\hat{\boldsymbol{a}}'_j\}, \hat{\boldsymbol{c}})$ in Game 0 and in Game 1 are statistically indistinguishable.

First notice that, by Lemma 3, we have that the following two distributions are statistically indistinguishable for every fixed $\hat{\boldsymbol{b}}^\star$ and $\hat{\boldsymbol{d}}_j^\star$:

$$\begin{aligned}(\boldsymbol{a}_i, \boldsymbol{a}'_{i,j}, [\boldsymbol{x}_i|\boldsymbol{x}_i R_{i,j}^\star]) \approx_s \\ (\boldsymbol{a}_i, \boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star, [\boldsymbol{x}_i|\boldsymbol{x}_i R_{i,j}^\star]) .\end{aligned}$$

Since $[\mathsf{rot}_f(\boldsymbol{a}_i)|\mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star)]^\top \boldsymbol{s}$ is statistically close to $[\mathsf{rot}_f(\boldsymbol{a}_i)|\mathsf{rot}_f(\boldsymbol{a}'_{i,j})]^\top \boldsymbol{s}$, it is possible to add each term to one side of the equation:

$$\begin{aligned}(\boldsymbol{a}_i, \boldsymbol{a}'_{i,j}, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i|(\mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star))^\top \boldsymbol{s} + \boldsymbol{x}_i R_{i,j}^\star]) \approx_s \\ (\boldsymbol{a}_i, \boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i|\mathsf{rot}_f(\boldsymbol{a}'_{i,j})^\top \boldsymbol{s} + \boldsymbol{x}_i R_{i,j}^\star]) .\end{aligned}$$

Finally, we add $\mathsf{rot}_f(\boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star)^\top \boldsymbol{s}$ to both sides:

$$\begin{aligned}(\boldsymbol{a}_i, \boldsymbol{a}'_{i,j}, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i|(\mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star))^\top \boldsymbol{s} + \boldsymbol{x}_i R_{i,j}^\star]) \approx_s \\ (\boldsymbol{a}_i, \boldsymbol{a}_i \otimes \boldsymbol{r}_{i,j}^\star - \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star, [\mathsf{rot}_f(\boldsymbol{a}_i)^\top \boldsymbol{s} + \boldsymbol{x}_i|\mathsf{rot}_f(\boldsymbol{a}'_{i,j} + \boldsymbol{b}_i^\star \otimes \boldsymbol{d}_{i,j}^\star)^\top \boldsymbol{s} + \boldsymbol{x}_i R_{i,j}^\star]) .\end{aligned}$$

To conclude, observe that if we concatenate all the parts, we have that the distribution on the left hand side is that of the public parameters and the challenge ciphertext in Game 0, while that on the right hand side is the distribution in Game 1.

□

4.3.2 Indistinguishability of Game 1 and Game 2 (or Game 3 and Game 4)

Lemma 12. *The view of adversary $\mathcal{A}$ in Game 1 (resp. Game 3) is computationally indistinguishable from the view of $\mathcal{A}$ in Game 2 (resp. Game 4) under decision-*RingLWE.

Proof.

Suppose $\mathcal{A}$ can distinguish between Game 1 and Game 2 with non-negligible advantage. Then, it is possible to use $\mathcal{A}$ to build an algorithm $\mathcal{B}$ to solve decision-RingLWE.

Init $\mathcal{B}$ is given k RingLWE challenge pairs $(\boldsymbol{a}_j, \boldsymbol{y}_j) \in \mathcal{R}_q \times \mathcal{R}_q$, where either $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$ for a random $\boldsymbol{s} \in \mathcal{R}_q$ and a noise term $\boldsymbol{x}_j \leftarrow \Psi_\alpha^n$, or $\boldsymbol{y}_j$ is uniformly random in $\mathcal{R}_q$. And one LWE challenge pair $(\boldsymbol{a}_0, y_0) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, where either $y_0 = \langle \boldsymbol{a}_0, \boldsymbol{s} \rangle + x_0$ for a noise term $x_0 \leftarrow \Psi_\alpha$, or y_0 is uniformly random in $\mathbb{Z}_q$.

SetUp The public parameters are constructed using the vectors of the pairs $(\boldsymbol{a}_j, \boldsymbol{y}_j)$. The i-th polynomial of ring $\hat{\boldsymbol{a}}$ will be the vector $\boldsymbol{a}_i$, for $1 \leq i \leq k$ and vector $\boldsymbol{u}$ will be $\boldsymbol{a}_0$. The rings $\hat{\boldsymbol{a}}'_j$ are still calculated as in Sim.SetUp, i.e., $\boldsymbol{a}'_{i,j} \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j} - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}$.

Secret keys All private-key extraction queries are answered using Sim.KeyGen.

Challenge Ciphertext The ciphertext $CT = (\hat{\boldsymbol{c}}^\star, c'^\star)$ is constructed based on the LWE challenge pairs $(\boldsymbol{a}_j, \boldsymbol{y}_j)$, with $\hat{\boldsymbol{c}}^\star = [(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) | R_0^{\star\top}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m) | \cdots | R_t^{\star\top}(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_m)]$ and $c'^\star = y_0 + M\lfloor q/2 \rceil$. If we have $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$ on the RingLWE challenge and $y_0 = \langle \boldsymbol{a}_0, \boldsymbol{s} \rangle + x_0$ on the LWE challenge, then the ciphertext is distributed exactly as in Game 1, and if $\boldsymbol{y}_j$ is uniformly random in $\mathcal{R}_q$ and y_0 is uniformly random in $\mathbb{Z}_q$, then the ciphertext is distributed exactly as in Game 2.

If $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$, then $(y_1, \ldots, y_m) = (\boldsymbol{a}_1 \otimes \boldsymbol{s} + \boldsymbol{x}_1, \ldots, \boldsymbol{a}_k \otimes \boldsymbol{s} + \boldsymbol{x}_m) = \mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}$.

Therefore, for Game 1 we have

$$
\begin{aligned}
C_{i,j} &= \mathsf{rot}_f(\boldsymbol{a}'_{i,j}) + \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}_{i,j}) \\
&= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j} - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}) + \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}_{i,j}) \\
&= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j}) - \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}) + \mathsf{rot}_f(\boldsymbol{b}^\star_i \otimes \boldsymbol{d}_{i,j}) \\
&= \mathsf{rot}_f(\boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j}) \\
&= \mathsf{rot}_f(\boldsymbol{a}_i)\mathsf{rot}_f(\boldsymbol{r}^\star_{i,j}),
\end{aligned}
$$

$$
\begin{aligned}
C_j &= [C_{0,j} | \cdots | C_{k-1,j}] \\
&= [\mathsf{rot}_f(\boldsymbol{a}_0)\mathsf{rot}_f(\boldsymbol{r}^\star_{0,j}) | \cdots | \mathsf{rot}_f(\boldsymbol{a}_{k-1})\mathsf{rot}_f(\boldsymbol{r}^\star_{k-1,j})] \\
&= R_j^{\star\top}\mathsf{Rot}_f(\hat{\boldsymbol{a}})
\end{aligned}
$$

and

$$\begin{aligned}
\hat{\boldsymbol{c}} &= [A|C_1|\cdots|C_t]^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_1^\star|\cdots|\hat{\boldsymbol{x}}R_t^\star] \\
&= [\mathsf{Rot}_f(\hat{\boldsymbol{a}})|R_1^{\star\top}\mathsf{Rot}_f(\hat{\boldsymbol{a}})|\cdots|R_t^{\star\top}\mathsf{Rot}_f(\hat{\boldsymbol{a}})]^\top \boldsymbol{s} + [\hat{\boldsymbol{x}}|\hat{\boldsymbol{x}}R_1^\star|\cdots|\hat{\boldsymbol{x}}R_t^\star] \\
&= [\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}|R_1^{\star\top}\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}R_1^\star|\cdots|R_t^{\star\top}\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}R_t^\star] \\
&= [\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}}|R_1^{\star\top}(\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}})|\cdots|R_t^{\star\top}(\mathsf{Rot}_f(\hat{\boldsymbol{a}})^\top \boldsymbol{s} + \hat{\boldsymbol{x}})] \\
&= [(y_1,\ldots,y_m)|R_1^{\star\top}(y_1,\ldots,y_m)|\cdots|R_t^{\star\top}(y_1,\ldots,y_m)] \ .
\end{aligned}$$

If y_j is uniformly random in $\mathbb{Z}_q$ then the ciphertext is uniformly random, as the ciphertext generated by Game 2.

Guess $\mathcal{A}$ must guess whether it is interacting with Game 1 or Game 2. The answer to this guess is also the answer to the $\mathsf{RingLWE}$ and LWE challenges, because, as we showed, if $\boldsymbol{y}_j$ is uniformly random in $\mathcal{R}_q$ and y_0 is uniformly random in $\mathbb{Z}_q$, then $\mathcal{A}$'s view is the same as in Game 2 and if $y_0 = \langle \boldsymbol{a}_0, \boldsymbol{s}\rangle + x_0$ and $\boldsymbol{y}_j = \boldsymbol{a}_j \otimes \boldsymbol{s} + \boldsymbol{x}_j$, then $\mathcal{A}$'s view is the same as in Game 1.

□

4.3.3 Indistinguishability of Game 2 and Game 3

Lemma 13. *The view of adversary $\mathcal{A}$ in Game 2 is statistically indistinguishable from the view of $\mathcal{A}$ in Game 3*

Proof.

SetUp The public parameters are generated in the same way in both games. Vectors $\hat{\boldsymbol{a}}$ and $\boldsymbol{u}$ are random and $\boldsymbol{a}'_{i,j} \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j} - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}$, with $\hat{\boldsymbol{d}}^\star_j = \hat{\boldsymbol{d}}^0_j$ for Game 2 and $\hat{\boldsymbol{d}}^\star_j = \hat{\boldsymbol{d}}^1_j$ for Game 3.

Secret keys All private-key extraction queries are answered using $\mathsf{Sim.KeyGen}$. The only difference is, again, that for Game 2, $id^\star_j = id^0_j$ and, for Game 3, $id^\star_j = id^1_j$.

Challenge Ciphertext The challenge ciphertext in both games is randomly chosen.

All public parameters are randomly generated in both games, except for rings $\hat{\boldsymbol{a}}'_j$. Therefore, the indistinguishability of Game 2 and Game 3 only depends on the indistinguishability of $\hat{\boldsymbol{a}}'_j$. From Lemma 3 we can prove that $\hat{\boldsymbol{a}}'_j$ for each game is statistically close to a uniformly random ring, because

$$\boldsymbol{a}'_{i,j} \leftarrow \boldsymbol{a}_i \otimes \boldsymbol{r}^\star_{i,j} - \boldsymbol{b}^\star_i \otimes \boldsymbol{d}^\star_{i,j}.$$

□

5 Conclusion

In this paper we present an ideal lattice-based IBE scheme based on the one presented by Agrawal et al. [1]. The scheme presented by Okuhata et al. [16] is similar to ours, but it does not fully take advantage of using ideal lattices. Representing the lattice basis as a vector of length m instead of a $n \times m$ matrix, and using the function rot to create the matrices, makes it possible to reduce the public-key size. Beside that, we present a clearer

scheme and detailed proof of security for our scheme. Although really similar, our schemes were developed independently of the schemes described by Yang et al. [23]. Beside the ideal lattice IBE, we also provide the hierarchical ideal lattice version of the scheme.

The first main advantage of the new scheme is the decrease in public-key size. In the original scheme, it is $O(mn)$, but as we do use vectors instead of matrices, the size is decreased to $O(n)$. The private keys, master key and ciphertext have exactly the same size in both schemes. The second main advantage is the decrease in encryption complexity. Since the new basis is a circular (Toeplitz) matrix, it allows fast multiplications [17]. Note that as the TrapGen algorithm, the IdealTrapGen algorithm is also polynomial and functions rot, Rot and Exp do not affect the complexity. Therefore, the remain algorithms in the scheme have the same complexity as the original ones.

As in the Agrawal et al. IBE scheme, our new scheme is shown to be weak selective secure based on the difficulty of a special case of the Learning With Errors Problem (LWE) for ideal lattices. Therefore, our scheme retains the security of the original scheme with the advantage of decreasing the public-key size and the complexity of the encryption algorithm.

References

[1] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572, French Riviera, 2010. springer.

[2] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40, Seoul, South Korea, 2011. springer.

[3] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Annual ACM Symposium on Theory of Computing*, pages 99–108, Philadephia, Pennsylvania, USA, 1996. ACM Press.

[4] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In *STACS 2009*, pages 75–86, 2009.

[5] Dan Boneh and Matthew K. Franklin. Identity based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[6] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552, French Riviera, 2010. springer.

[7] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA 2001*, volume 2260 of *LNCS*, pages 360–363, Cirencester, UK, 2001. springer.

[8] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206, Victoria, British Columbia, Canada, 2008. ACM Press.

[9] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, pages 89–98, Alexandria, Virginia, USA, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309.

[10] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162, Istanbul, Turkey, 2008. springer.

[11] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera, 2010. springer.

[12] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010.

[13] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions from worst-case complexity assumptions. In *FOCS 2002*, pages 356–365, Vancouver, British Columbia, Canada, 2002. IEEE.

[14] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on Gaussian measures. In *FOCS 2004*, pages 372–381, Rome, Italy, 2004. IEEE.

[15] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 214–231, Tokyo, Japan, 2009. springer.

[16] Ryouta Okuhata, Yoshifumi Manabe, and Tatsuaki Okamoto. An identity based encryption scheme from ideal lattices. In *SCIS 2011*, 2011.

[17] Victor Y. Pan. *Structured matrices and polynomials: unified superfast algorithms.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[18] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *TCC 2006*, volume 3876 of *LNCS*, pages 145–166, New York, NY, USA, 2006. springer.

[19] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pages 84–93, Baltimore, Maryland, USA, 2005. ACM Press.

[20] Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473, Aarhus, Denmark, 2005. springer.

[21] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO 1984*, volume 196 of *LNCS*, pages 47–53, Santa Barbara, CA, USA, 1985. springer.

[22] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*, volume 5479 of *LNCS*, pages 617–635, Cologne, Germany, 2009. springer.

[23] Xiao yuan Yang, Li qiang Wu, Min qing Zhang, and Xiao-Feng Chen. An efficient CCA-secure cryptosystem over ideal lattices from identity-based encryption. *Computers and Mathematics with Applications*, pages 1254–1263, 2013.