Expanding a Lattice-based HVE Scheme

Karina Mochetti[1], **Ricardo Dahab**[1]

[1]Instituto de Computação (UNICAMP)
Av. Albert Einstein, 1251, 13083-852, Campinas-SP, Brazil

{mochetti,rdahab}@ic.unicamp.br

Abstract. *Functional encryption systems provide finer access to encrypted data by allowing users to learn functions of encrypted data. A Hidden-Vector Encryption Scheme (HVE) is a functional encryption primitive in which the ciphertext is associated with a binary vector $\boldsymbol{w}$ and the secret key is associated with a special binary vector $\boldsymbol{v}$ that allows "don't care" entries. The decryption is only possible if the vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ are the same for all elements, except the "don't care" entries in $\boldsymbol{v}$. HVE schemes are used to construct more sophisticated schemes that support conjunctive and range searches. In this work we show how to expand the basic fuzzy IBE scheme of Agrawal et al. (PKC 2012) to a hierarchical HVE scheme. We also show how the version using ideal lattices affects the security proof.*

1. Introduction

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, thus providing a much finer control of decryption capabilities. Some functional encryption primitives are Identity-Based Encryption (IBE), Attribute-Based Encryption (ABE), Inner-Product Encryption (IPE) and Hidden-Vector Encryption (HVE) [Boneh et al. 2011] .

In a Hidden-Vector Encryption (HVE) scheme the ciphertext is associated with a binary vector $\boldsymbol{w}$ and the secret key is associated with a special binary vector $\boldsymbol{v}$ that allows "don't care" entries (denoted by $\star$). The decryption is only possible if the vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ are the same for all elements that are not represented by $\star$ in vector $\boldsymbol{v}$.

HVE schemes are used on more sophisticated functional encryption schemes that support conjunctive and range searches, for example. The first scheme was proposed in [Boneh and Waters 2007] based on bilinear groups and proved secure under the selective model. Other schemes, also based on bilinear groups, were developed, such as the scheme proposed in [Iovino and Persiano 2008] that uses bilinear groups of prime order and the scheme presented in [Caro et al. 2011] that is the first fully secure construction known.

Most lattice-based HVE schemes known can be built from Inner Product Encryption, such as [Agrawal et al. 2011] and [Abdalla et al. 2012]. Such construction was first presented in [Katz et al. 2008]. Also, the basic fuzzy IBE scheme presented in [Agrawal et al. 2012] can clearly be seen as an HVE scheme. The main focus of this work is to expand the underlying scheme to an ideal lattice-based HVE scheme and to a hierarchical HVE scheme.

Ideal lattices are a generalization of cyclic lattices, first presented in [Micciancio 2002], in which the lattice corresponds to ideals in a ring $\mathbb{Z}[x]/\langle f(x)\rangle$, for some irreducible polynomial function $f(x)$. 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.

For cryptosystems that use a Trusted Third Part, as HVE schemes, it is convenient to have a hierarchy of certificate authorities, that is, the root certificate authority can issue certificates for other certificate authorities, which can issue certificates for users. A scheme in which a user in level t can use his/her secret key to derive a secret key for a user at level $t+1$ is called hierarchical, as introduced in [Hanaoka et al. 2009]. This reduces the workload on a Thrusted Third Part as it does not need to generate all public and master keys.

Our Contributions. In this work we show how to expand the basic fuzzy IBE scheme of [Agrawal et al. 2012] to a hierarchical HVE scheme. We show that this expansion has the same security, based on the Learning With Errors Problem (LWE), as the original scheme, while having the new feature which reduces the role of the Thrusted Third Part in the scheme. We also show a version using ideal lattices, that has the advantages of smaller key sizes and more efficient matrix multiplication, and how it affects the security proof.

2. Definitions

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. $\boldsymbol{A}$) to denote matrices, bold lowercase letters (e.g. $\boldsymbol{w}$) to denote vectors. The notation $\boldsymbol{A}^\top$ denotes the transpose of matrix $\boldsymbol{A}$. 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 $\boldsymbol{A}_1$ is an $n \times m$ matrix and $\boldsymbol{A}_2$ is an $n \times m'$ matrix, then $[\boldsymbol{A}_1|\boldsymbol{A}_2]$ denotes the $n \times (m+m')$ matrix formed by concatenating the columns of $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$. 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 $|\boldsymbol{v}| = \sqrt{\sum \boldsymbol{x}_i^2}$ as the norm of vector $\boldsymbol{v}$, and for matrix $\boldsymbol{A}$, we define $|\boldsymbol{A}| = \max|\boldsymbol{A}\boldsymbol{x}|$, for $|\boldsymbol{x}| = 1$, as the norm of matrix $\boldsymbol{A}$. We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\mathrm{negl}(n)$ to denote a negligible function of n. We say that function $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use $\mathrm{poly}(n)$ to denote a polynomial function of n. We say an event occurs with *overwhelming probability* if its probability is $1 - \mathrm{negl}(n)$. 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 $f(x)$. The notation $[d]$ denotes the set of positive integers $\{1, 2, \ldots, d\}$.

2.1. Hidden Vector Encryption

Based on the definition of predicate encryption by [Katz et al. 2008], we have that a Hidden Vector Encryption Scheme consists of the following four algorithms:

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

KeyGen(mpk, msk, v). Takes as input public-key mpk, master secret key msk and a vector $\boldsymbol{v} \in \{0,1,\star\}^l$ and outputs a secret key sk.

Enc($mpk, \mathfrak{m}, w$). Takes as input public parameters, message $\mathfrak{m}$ from some associated message space, public-key mpk, a vector $\boldsymbol{w} \in \{0,1\}^l$ and outputs a ciphertext C.

Dec(mpk, sk, C). Takes as input public-key mpk, ciphertext C, secret key sk and outputs the message $\mathfrak{m}$.

Suppose ciphertext C is obtained by running Enc on input mpk, message $\mathfrak{m}$ and vector $\boldsymbol{w}$ and that sk is a secret key obtained through a call of KeyGen using the same mpk and vector $\boldsymbol{v}$. Then Dec, on input mpk, C and sk returns $\mathfrak{m}$, except with negligible probability, if and only if $v_i = w_i$, for all $i \in [l]$ such that $v_i \neq \star$.

For a hierarchical scheme, the KeyGen algorithm is replaced by the Derive algorithm:

Derive(mpk, sk_{t-1}, v). Takes as input public-key mpk, secret key sk_{t-1} for hierarchical level $t-1$ and a vector $\boldsymbol{v} \in \{0,1,\star\}^l$, and outputs a secret key sk_t for level t.

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

Init. $\mathcal{A}$ outputs challenge vectors $\boldsymbol{w}_0^\star, \boldsymbol{w}_1^\star$.

Setup. The challenger $\mathcal{B}$ runs the Setup algorithm to generate public-key mpk which it gives to the adversary $\mathcal{A}$.

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

Challenge. The adversary $\mathcal{A}$ gives a pair of messages $(\mathfrak{m}_0, \mathfrak{m}_1)$ to the challenger $\mathcal{B}$. Then $\mathcal{B}$ chooses random $\eta \xleftarrow{\$} \{0,1\}$, encrypts $\mathfrak{m}_\eta$ under $\boldsymbol{w}_\eta$ 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 $|\Pr[\eta' = \eta] - \frac{1}{2}|$, then we say the scheme has indistinguishability under chosen-plaintext attack under the selective model (IND-sAT-CPA for short).

2.2. Lattices

This section presents the collection of results from [Agrawal et al. 2011, Lyubashevsky et al. 2010, Micciancio and Regev 2004, Cash et al. 2010] 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 $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, we define

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

2.2.1. Ideal Lattices

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 $\Lambda_q^{\perp}(\boldsymbol{A})$, with $\boldsymbol{A} = \mathsf{rot}_f(g) \in \mathbb{Z}_q^{n \times n}$, where each row i of $\boldsymbol{A}$ 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 $\boldsymbol{A} = \mathsf{rot}_f(g) \in \mathbb{Z}_q^{n \times n}$ is an anti-circulant matrix and for $f(x) = x^n - 1$, we have that the matrix $\boldsymbol{A} = \mathsf{rot}_f(g) \in \mathbb{Z}_q^{n \times n}$ is a circulant matrix. These lattices are called cyclic lattices and they are a special class of ideal lattices. 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 [Pan 2001].

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

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

2.2.2. Sampling Algorithms

This section gives the main definitions and theorems over lattices used to generate trapdoor functions.

Definition 1. *Let $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k\}$ be a set of vectors in $\mathbb{R}^m$. Let $|\boldsymbol{S}|$ denotes the length of the longest vector in $\boldsymbol{S}$, i.e., $\max_{1 \leq i \leq k} |\boldsymbol{s}_i|$, and $\widetilde{\boldsymbol{S}} := \widetilde{\boldsymbol{s}}_1, \ldots, \widetilde{\boldsymbol{s}}_k \subset \mathbb{R}^m$ denotes the* Gram-Schmidt orthogonalization *of the vectors $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_k$. We refer to $|\widetilde{\boldsymbol{S}}|$ as the* Gram-Schmidt norm *of $\boldsymbol{S}$.*

Definition 2. *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 $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w}) := \exp\left(-\pi|x - \boldsymbol{c}|^2/\sigma^2\right)$ be the Gaussian function on $\mathbb{R}^n$ with center $\boldsymbol{c}$ and parameter σ. Let $\rho_{\sigma,\boldsymbol{c}}(L) := \sum_{\boldsymbol{w} \in L} \rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w})$ 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 theorem shows how to sample an essentially uniform matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ along with a basis $\boldsymbol{S}$ of $\Lambda_q^\perp(\boldsymbol{A})$ with low Gram-Schmidt norm.

Theorem 1. *[Alwen and Peikert 2009] 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 $(\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}, \boldsymbol{S} \in \mathbb{Z}^{m\times m})$ such that $\boldsymbol{A}$ is statistically close to uniform in $\mathbb{Z}_q^{n\times m}$ and $\boldsymbol{S}$ is a basis for $\Lambda_q^\perp(\boldsymbol{A})$, satisfying $|\widetilde{\boldsymbol{S}}| \leq O(\sqrt{n\log q})$ and $|\boldsymbol{S}| \leq O(n\log q)$ with overwhelming probability in n.*

The following theorem shows an adaptation of Ajtai's trapdoor key generation algorithm for ideal lattices.

Theorem 2. *[Stehlé et al. 2009] 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 $(\vec{a} \in \mathcal{R}^k, \boldsymbol{S} \in \mathbb{Z}^{kn\times kn})$ such that $\vec{a}$ is statistically close to uniform in $\mathcal{R}^k$ and $\boldsymbol{S}$ is a basis for $\Lambda_q^\perp(\boldsymbol{A})$, for $\boldsymbol{A} = \mathsf{Rot}_f(\vec{a})$, satisfying $|\boldsymbol{S}| = O(n\log q\sqrt{\omega(\log n)})$ with overwhelming probability in n.*

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

Theorem 3. *[Gentry et al. 2008] Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ be a full rank matrix, let $\boldsymbol{S}$ be a short basis of $\Lambda_q^\perp(\boldsymbol{A})$ and let σ be the Gaussian parameters. For q, m, n integers such that $q > 2$ and $m > n$ and $\sigma > \|\boldsymbol{S}\|\cdot\omega(\sqrt{\log m})$, there is a probabilistic polynomial algorithm $\mathsf{SamplePre}(\boldsymbol{A}, \boldsymbol{S}, \boldsymbol{u}, \sigma)$ that outputs a vector $\boldsymbol{e} \in \mathbb{Z}^m$ statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}$.*

Theorem 4. *[Gentry et al. 2008] Let $\sigma, \vec{c}$ be Gaussian parameter, let $\boldsymbol{A} \in \mathbb{Z}^{n\times m}$ be a matrix and let n, m be integers such that $\sigma \geq \|\boldsymbol{A}\| \cdot \omega(\sqrt{\log n})$. Then there is a probabilistic polynomial algorithm $\mathsf{SampleGaussian}(\boldsymbol{A}, \sigma)$ that outputs a vector $\boldsymbol{e} \in \mathbb{Z}^m$ statistically close to $\mathcal{D}_{\Lambda(\boldsymbol{A}),\sigma}$.*

Theorem 5. *[Agrawal et al. 2010] Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ be a full rank matrix, let $\boldsymbol{S}$ be a short basis of $\Lambda_q^\perp(\boldsymbol{A})$, let $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m_1}$ 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 $\sigma > \|\boldsymbol{S}\| \cdot \omega(\sqrt{\log(m + m_1)})$, there is a probabilistic polynomial algorithm $\mathsf{SampleBasisLeft}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{S}, \sigma)$ that outputs a new basis $\boldsymbol{T} \in \mathbb{Z}^{n\times m+m_1}$ for lattice $\Lambda_q^\perp(\boldsymbol{F})$, with $\boldsymbol{F} = (\boldsymbol{A}|\boldsymbol{B})$.*

Theorem 6 ([Agrawal et al. 2010])**.** *Let $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m_1}$ be full rank matrices, let $\boldsymbol{S}$ be a short basis for $\Lambda_q^\perp(\boldsymbol{B})$, let $\boldsymbol{R} \in \{-1, 1\}^{m\times m1}$ 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 $\sigma > \|\boldsymbol{S}\| \cdot \sqrt{m} \cdot \omega(\sqrt{\log m})$. Then there is a probabilistic polynomial algorithm $\mathsf{SampleBasisRight}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{S}, \sigma)$ that outputs a new basis $\boldsymbol{T} \in \mathbb{Z}^{n\times m+m_1}$ for lattice $\Lambda_q^\perp(\boldsymbol{F})$, with $\boldsymbol{F} = (\boldsymbol{A}|\boldsymbol{A}\boldsymbol{R} + \boldsymbol{B})$.*

2.2.3. Learning With Errors 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 3. *[Regev 2005] 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 $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{r} \rangle + e)$. 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 $\mathbb{Z}_q^n \times Z_q$, output 0 if the former holds and 1 if the latter holds.*

The hardness of the LWE problem is summarized in the following

Definition 4. *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. *[Regev 2005] 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* GapSVP *to within $\widetilde{O}(n/\alpha)$ in the worst case.*

The ideal-LWE Problem is the same as described above, but with $\boldsymbol{a}$ chosen uniformly in $\mathbb{Z}_q[x]/f(x)$. The hardness of the ideal-LWE problem is summarized in the following:

Theorem 8 ([Lyubashevsky et al. 2010])**.** *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-ideal-$\mathsf{LWE}_{q,n,\Upsilon_\alpha}$*, then there exists an efficient quantum algorithm that solves γ-SIVP and γ-SVP for $\gamma = \widetilde{O}(n/\alpha)$ in the worst case.*

2.3. Shamir's Secret Sharing

Shamir's Secret Sharing [Shamir 1979] is a threshold scheme, i.e., a scheme to divide data into n parts in a way that it is only possible to recover the data with at least r parts, for $r \leq n$. The scheme is based on polynomial interpolation, i.e., for data d we create shares by choosing a random polynomial p of degree $r-1$ with $p(0) = d$, then each share piece d_i, for $i \in [n]$ will be a point defined by the polynomial, so $d_i = p(i)$

To recover the data, the polynomial is rebuilt using r points. Several algorithms for polynomial evaluation and interpolation are known and can be used. One of the most efficient method known is the Lagrange Algorithm, which calculates r polynomials $l_j(x)$, called Lagrangian coefficients, based on the r given points $(x_j, y_j) = (i, d_i)$ and reconstruct the polynomial $p(x)$ calculating

$$p(x) = \sum_{j=0}^{k} y_j l_j(x)$$

where,

$$l_j(x) = \prod_{m=0}^{k} \frac{x - x_m}{x_j - x_m}.$$

The data will, therefore, be $d = p(0) = \sum_{j=0}^{k} y_j l_j(0)$.

Lemma 1. *([Agrawal et al. 2012, Lemma 3]) Let $\beta = (l!)^2$. Given $k \leq l$ numbers $x_1, \cdots, x_k \in [1, l]$ define the lagrangian coefficients*

$$l_j = \prod_{i \neq j} \frac{-x_i}{x_j - x_i}.$$

Then, for every $1 \leq j \leq k$, the value βl_j is an integer, and $|\beta l_j| \leq \beta^2 \leq (l!)^4$.

3. Hierarchical Lattice-Based HVE Scheme

In this section we provide our hierarchical HVE scheme, with its correctness and security proof.

3.1. Our Construction

Let n be the security parameter, σ be the Gaussian parameter, l be the length of vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, r the threshold value and h the hierarchical depth.

$\mathsf{Setup}(1^n)$. On input of security parameter n, the algorithm generates the public and secret keys as follows:

(i) run the $\mathsf{TrapGen}(n, q, \sigma)$ algorithm to select uniformly l matrices $\boldsymbol{A}_i \in \mathbb{Z}^{n \times m}$ (for $i \in [l]$), with a short basis $\boldsymbol{T}_{\boldsymbol{A}_i} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^{\perp}(\boldsymbol{A}_i)$;
(ii) choose uniformly random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$;
(iii) choose random matrices $\boldsymbol{A}_{i,b,j}$ and $\boldsymbol{B}_i$ (for $i \in [l]$, $b \in \{0, 1\}$ and $j \in [h]$);
(iv) output $mpk = (\{\boldsymbol{A}_i\}, \boldsymbol{u}, \{\boldsymbol{A}_{i,b,j}\}, \{\boldsymbol{B}_i\})$ and $msk = \{\boldsymbol{T}_{\boldsymbol{A}_i}\}$.

$\mathsf{Derive}(\boldsymbol{mpk}, \boldsymbol{sk_{t-1}}, \boldsymbol{v_1}, \cdots, \boldsymbol{v_t})$. On input of public-key mpk, secret key for the hierarchical level $t-1$ and vectors $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_t$, the algorithm generates a secret key sk_t as follows:

(i) sample a new short basis for each lattice $\Lambda(\boldsymbol{F}_i || \boldsymbol{A}_{i,v_{t,i},t} + \boldsymbol{B}_i)_q^{\boldsymbol{u}}$, for $\boldsymbol{F}_i = [\boldsymbol{A}_i || \boldsymbol{A}_{i,v_{1,i},1} + \boldsymbol{B}_i || ... || \boldsymbol{A}_{i,v_{t-1,i},t-1} + \boldsymbol{B}_i]$ by involving $\boldsymbol{S}_i \leftarrow \mathsf{SampleBasisLeft}(\boldsymbol{F}_i, \boldsymbol{A}_{i,v_{t,i},t} + \boldsymbol{B}_i, \boldsymbol{S}_i', \sigma)$, where $\boldsymbol{S}_i' \in sk_{t-1}$;
(ii) output $sk_t = \{\boldsymbol{S}_i\}$.

$\mathsf{Enc}(\boldsymbol{mpk}, \mathfrak{m}, \boldsymbol{w_1}, \cdots, \boldsymbol{w_t})$. On input of master public-key mpk, vectors $\boldsymbol{w}_1, \cdots, \boldsymbol{w}_t$, and message $\mathfrak{m} \in \{0, 1\}$, the algorithm generates a ciphertext C as follows:

(i) choose a uniformly random vector $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, random matrices $\boldsymbol{R}_{i,j} \in \{-1, 1\}^{m \times m}$ and let $\beta = (l!)^2$;

(ii) choose a noise vector $\boldsymbol{x}_i \leftarrow \overline{\Psi}_{\alpha_t}^m$ and a noise term $x \leftarrow \overline{\Psi}_{\alpha_t}$;
(iii) calculate $\boldsymbol{F}_i = [\boldsymbol{A}_i || \boldsymbol{A}_{i,v_{1,i},1} + \boldsymbol{B}_i || ... || \boldsymbol{A}_{i,v_{t,i},t} + \boldsymbol{B}_i]$;
(iv) create l shares of vector $\boldsymbol{s}$ such that $\boldsymbol{s}_i = [p_1(i), ..., p_n(i)]$, for n random polynomials $p_i(x)$ of degree $r-1$, with $p_i(0) = s_i$
(v) compute $\boldsymbol{c}_i = \boldsymbol{F}_i^\top \boldsymbol{s}_i + \beta[\boldsymbol{x}_i || \boldsymbol{R}_i^\top \boldsymbol{x}_i] \in \mathbb{Z}_q^{(t+1)m}$, where $\boldsymbol{R}_i = [\boldsymbol{R}_{i,1} || ... || \boldsymbol{R}_{i,t}]$ and $c' = \boldsymbol{u}^\top \boldsymbol{s} + \beta \cdot x + \mathfrak{m} \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q$;
(vi) output $C = (\{\boldsymbol{c}_i\}, c')$.

$\mathsf{Dec}(\boldsymbol{mpk}, \boldsymbol{sk_t}, \boldsymbol{C})$. On input of master public key mpk, secret key sk_t, and ciphertext C, the algorithm does the following:

(i) let $\mathbb{J}$ be the set of matching bits where $\boldsymbol{v}_{\gamma,i} = \boldsymbol{w}_{\gamma,i}$ for all $\gamma \in [t]$ and calculate each polynomial $l_j(x) = \prod_{i \in \mathbb{J}} \frac{x-i}{j-i}$;
(ii) the fractional Lagrangian coefficients will be $l_j = l_j(0)$ for $j \in \mathbb{J}$ so that $\sum l_j \boldsymbol{u}^\top \boldsymbol{s}_j = \boldsymbol{u}^\top \boldsymbol{s}$;
(iii) calculate each $\boldsymbol{e}_j$ by calling $\mathsf{SamplePre}(\boldsymbol{F}_j, \boldsymbol{S}_j, l_j \boldsymbol{u}, \sigma)$, where $\boldsymbol{S}_j \in sk_t$ and $\sigma = \sigma_t \sqrt{m(t+1)} \omega(\sqrt{\log(tm)})$;
(iv) compute $z = c' - \sum \boldsymbol{e}_j^\top \boldsymbol{c}_j \pmod q$; for $z \in (-q/2, q/2]$, output 0 if $|z| < q/4$ and 1 otherwise. Note that the threshold now is done between all vectors hierarchical, i.e., r must be the matching lines between matrices $\boldsymbol{V} = [\boldsymbol{v}_1^\top | ... | \boldsymbol{v}_t^\top]$ and $\boldsymbol{W} = [\boldsymbol{w}_1^\top | ... | \boldsymbol{w}_t^\top]$.

3.2. Correctness

If $v_{\gamma,i} = w_{\gamma,i}$, for all $v_{\gamma,i} \neq \star$, we have:

$$
\begin{aligned}
z &= c' - \sum \boldsymbol{e}_j^\top \boldsymbol{c}_j \pmod q \\
&= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \sum \boldsymbol{e}_j^\top (\boldsymbol{F}_j^\top \boldsymbol{s}_j + \beta[\boldsymbol{x}_j || \boldsymbol{R}_j^\top \boldsymbol{x}_j]) \pmod q \\
&= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \sum (\boldsymbol{F}_j \boldsymbol{e}_j)^\top \boldsymbol{s}_j - \sum \boldsymbol{e}_j^\top \beta[\boldsymbol{x}_j || \boldsymbol{R}_j^\top \boldsymbol{x}_j] \pmod q \\
&= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \sum l_j \boldsymbol{u}^\top \boldsymbol{s}_j - \sum \boldsymbol{e}_j^\top \beta[\boldsymbol{x}_j || \boldsymbol{R}_j^\top \boldsymbol{x}_j] \pmod q \\
&= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \boldsymbol{u}^\top \boldsymbol{s} - \sum \boldsymbol{e}_j^\top \beta[\boldsymbol{x}_j || \boldsymbol{R}_j^\top \boldsymbol{x}_j] \pmod q \\
&= \mathfrak{m} \cdot \lfloor q/2 \rceil + \underbrace{\beta \cdot x - \sum \boldsymbol{e}_j^\top \beta[\boldsymbol{x}_j || \boldsymbol{R}_j^\top \boldsymbol{x}_j]}_{\text{error term}} \pmod q
\end{aligned}
$$

3.3. Security Reduction

In this section we prove the following theorem.

Theorem 9. *If the decision-*$\mathsf{LWE}_{q,n,\chi}$ *problem is infeasible, then the HVE scheme described on Section 3.1 is IND-sAT-CPA .*

Proof.

Init. $\mathcal{B}$ is given lm LWE challenge pairs $(\boldsymbol{y}_{i,j}, z_{i.j}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ and a pair $(\boldsymbol{y}, z) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where, either $[z_{i,1}|...|z_{i,m}] = [\boldsymbol{y}_{i,1}|...|\boldsymbol{y}_{i,m}]^\top \boldsymbol{s} + \beta \boldsymbol{x}_i$ and $z = \langle \boldsymbol{y}, \boldsymbol{s} \rangle + \beta x$ for a random $\boldsymbol{s} \in \mathbb{Z}_q^n$, and noise terms $x \leftarrow \overline{\Psi}_{\alpha_t}$ and $\boldsymbol{x}_i \leftarrow \overline{\Psi}_{\alpha_t}^m$, or $z_{i,j}$ and z are uniformly random.

Setup. The public key is constructed using the vectors of the challenge pairs, as follows: $\boldsymbol{A}_i$ will be $\beta\boldsymbol{Y}$, where $\boldsymbol{Y} = [\boldsymbol{y}_{i,1}|...|\boldsymbol{y}_{i,m}]$, $\boldsymbol{A}_{i,b,j}$ will be $\boldsymbol{A}_i\boldsymbol{R}_{i,j} - \boldsymbol{B}_i$, each $\boldsymbol{R}_{i,j}$ is random in $\{-1,1\}^m$, matrices $\boldsymbol{B}_i$ will be computed using $\mathsf{TrapGen}$ and $\boldsymbol{u}$ will be $\boldsymbol{y}$.

Secret keys. All private-key extraction queries are answered by using the trapdoor $\boldsymbol{T}_{\boldsymbol{B}_i}$ and the $\mathsf{SampleBasisRight}$ algorithm. It will output $sk_t = \{\boldsymbol{S}_i\}$, where $\boldsymbol{S}_i$ is a short basis for $\Lambda_q^{\perp}(\boldsymbol{A}_i\|\boldsymbol{A}_i\boldsymbol{R}_i + \boldsymbol{B}_i)$ by invoking

$$\boldsymbol{S}_i \leftarrow \mathsf{SampleBasisRight}(\boldsymbol{A}_i, \boldsymbol{B}_i, \boldsymbol{R}_i, \boldsymbol{T}_{\boldsymbol{B}_i}, \boldsymbol{u}, \sigma).$$

Note that the distribution of the public parameters and keys in the real scheme is statistically indistinguishable from that in the simulation, as in [Agrawal et al. 2010] and [Abdalla et al. 2012].

Challenge Ciphertext. The algorithm now chooses a $\boldsymbol{s}'^{\star} \in \mathbb{Z}_q^n$ at random and calculates $\boldsymbol{s}^{\star}$ such that all shares $\boldsymbol{s}_i^{\star} = \boldsymbol{s}'^{\star}$, i.e., it solves the equation $\boldsymbol{s}_i^{\star} = \boldsymbol{s} + \sum \boldsymbol{a}_j i^j$ for $\boldsymbol{s}_i^{\star} = \boldsymbol{s}'^{\star}$, $i \in [1,l]$ and $j \in [0, r-1]$.

The construction of ciphertext $C = (\{\boldsymbol{c}_i\}, c')$ is based on the terms of the LWE challenges pairs, $c' = \beta w + \mathfrak{m}\lfloor q/2 \rceil$ and $\boldsymbol{c}_i$ is a concatenation of vectors $\beta\boldsymbol{R}_{i,j}^{\top}[z_{i,0}|...|z_{i,m}]$. If $[z_{i,0}|...|z_{i,m}] = \boldsymbol{Y}^{\top}\boldsymbol{s}'^{\star} + \boldsymbol{x}_i$ and $w = \langle \boldsymbol{u}, \boldsymbol{s}\rangle + x$ on the LWE challenges, then we have that the ciphertext is genuine:

$$\boldsymbol{c}_i = \boldsymbol{F}_i^{\top}\boldsymbol{s}'^{\star} + \beta[\boldsymbol{x}_i||\boldsymbol{R}_i^{\top}\boldsymbol{x}_i]$$

$$\boldsymbol{c}_i = [\boldsymbol{A}_i||\boldsymbol{A}_{i,v_{1,i},1} + \boldsymbol{B}_i||...||\boldsymbol{A}_{i,v_{t,i},t} + \boldsymbol{B}_i]^{\top}\boldsymbol{s}'^{\star} + \beta[\boldsymbol{x}_i||\boldsymbol{R}_i^{\top}\boldsymbol{x}_i]$$

$$\boldsymbol{c}_i = [\boldsymbol{A}_i||\boldsymbol{A}_i\boldsymbol{R}_{i,1} - \boldsymbol{B}_i + \boldsymbol{B}_i||...||\boldsymbol{A}_i\boldsymbol{R}_{i,t} - \boldsymbol{B}_i + \boldsymbol{B}_i]^{\top}\boldsymbol{s}'^{\star} + \beta[\boldsymbol{x}_i||\boldsymbol{R}_i^{\top}\boldsymbol{x}_i]$$

$$\boldsymbol{c}_i = [\boldsymbol{A}_i||\boldsymbol{A}_i\boldsymbol{R}_{i,1}||...||\boldsymbol{A}_i\boldsymbol{R}_{i,t}]^{\top}\boldsymbol{s}'^{\star} + \beta[\boldsymbol{x}_i||\boldsymbol{R}_i^{\top}\boldsymbol{x}_i]$$

$$\boldsymbol{c}_i = [\boldsymbol{A}_i^{\top}\boldsymbol{s}'^{\star} + \beta\boldsymbol{x}_i||\boldsymbol{R}_{i,1}^{\top}(\boldsymbol{A}_i^{\top}\boldsymbol{s}'^{\star} + \beta\boldsymbol{x}_i)||...||\boldsymbol{R}_{i,t}^{\top}(\boldsymbol{A}_i^{\top}\boldsymbol{s}'^{\star} + \beta\boldsymbol{x}_i)]$$

$$\boldsymbol{c}_i = [\beta(\boldsymbol{Y}^{\top}\boldsymbol{s}'^{\star} + \boldsymbol{x}_i)||\beta\boldsymbol{R}_{i,1}^{\top}(\boldsymbol{Y}^{\top}\boldsymbol{s}'^{\star} + \boldsymbol{x}_i)||...||\beta\boldsymbol{R}_{i,t}^{\top}(\boldsymbol{Y}^{\top}\boldsymbol{s}'^{\star} + \boldsymbol{x}_i)]$$

If $z_{i,j}$ and z are uniformly random, then the ciphertext is randomly generated.

Guess. $\mathcal{A}$ must guess whether it is interacting with a genuine or with a randomly generated ciphertext. The answer to this guess is also the answer to one of the LWE challenges. We showed that if z and all $z_{i,j}$ are uniformly random, then the ciphertext is randomly generated and if $[z_{i,1}|...|z_{i,m}] = [\boldsymbol{y}_{i,1}|...|\boldsymbol{y}_{i,m}]^{\top}\boldsymbol{s} + \beta\boldsymbol{x}_i$ and $w = \langle \boldsymbol{y}, \boldsymbol{s}\rangle + \beta x$, then the ciphertext is genuine. Therefore, $\mathcal{B}$'s advantage in solving LWE is the same as $\mathcal{A}$'s advantage in distinguishing whether the ciphertext is genuine or not.

□

4. Conclusion

In this paper we expand the basic fuzzy IBE scheme proposed by [Agrawal et al. 2012] into a hierarchical HVE scheme, in which users can generate secret keys, thus relieving the task of the Trusted Third Part. Our scheme is as secure as the original one based on the Learning With Errors Problem, while having the new feature. We also show a version using ideal lattices, that has the advantages of smaller key sizes and more efficient matrix multiplication, and how it affects the security proof.

References

Abdalla, M., De Caro, A., and Mochetti, K. (2012). Lattice-based hierarchical inner product encryption. In *LATINCRYPT 2012*, volume 7533 of *LNCS*, pages 121–138, Santiago, Chile. springer.

Agrawal, S., Boneh, D., and Boyen, X. (2010). Efficient lattice (H)IBE in the standard model. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572, French Riviera. springer.

Agrawal, S., Boyen, X., Vaikuntanathan, V., Voulgaris, P., and Wee, H. (2012). Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC 2012*, volume 7293 of *LNCS*, pages 280–297, Darmstadt, Germany. springer.

Agrawal, S., Freeman, D. M., and Vaikuntanathan, V. (2011). Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40, Seoul, South Korea. springer.

Alwen, J. and Peikert, C. (2009). Generating shorter bases for hard random lattices. In *STACS 2009*, pages 75–86.

Boneh, D., Sahai, A., and Waters, B. (2011). Functional encryption: Definitions and challenges. In *TCC 2011*, volume 6597 of *LNCS*, pages 253–273, Providence, RI, USA. springer.

Boneh, D. and Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, volume 4392 of *LNCS*, pages 535–554, Amsterdam, The Netherlands. springer.

Caro, A. D., Iovino, V., and Persiano, G. (2011). Hidden vector encryption fully secure against unrestricted queries. *IACR Cryptology ePrint Archive*, 2011:546.

Cash, D., Hofheinz, D., Kiltz, E., and Peikert, C. (2010). Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552, French Riviera. springer.

Gentry, C., Peikert, C., and Vaikuntanathan, V. (2008). Trapdoors for hard lattices and new cryptographic constructions. In *STOC 2008*, pages 197–206, Victoria, British Columbia, Canada. ACM Press.

Hanaoka, G., Nishioka, T., Zheng, Y., and Imai, H. (2009). An efficient hierarchical identity-based key-sharing method resistant against collusion-attacks. In *ASIACRYPT 2009*, volume 5479 of *LNCS*, pages 348–362, Cologne, Germany. springer.

Iovino, V. and Persiano, G. (2008). Hidden-vector encryption with groups of prime order. In *PAIRING 2008*, volume 5209 of *LNCS*, pages 75–88, Egham, UK. springer.

Katz, J., Sahai, A., and Waters, B. (2008). Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162, Istanbul, Turkey. springer.

Lyubashevsky, V., Peikert, C., and Regev, O. (2010). On ideal lattices and learning with errors over rings. In *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23, French Riviera. springer.

Micciancio, D. (2002). 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. IEEE.

Micciancio, D. and Regev, O. (2004). Worst-case to average-case reductions based on Gaussian measures. In *FOCS 2004*, pages 372–381, Rome, Italy. IEEE.

Pan, V. Y. (2001). *Structured matrices and polynomials: unified superfast algorithms*. Springer-Verlag New York, Inc., New York, NY, USA.

Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. In *STOC 2005*, pages 84–93, Baltimore, Maryland, USA. ACM Press.

Shamir, A. (1979). How to share a secret. *Commun. ACM*, 22(11):612–613.

Stehlé, D., Steinfeld, R., Tanaka, K., and Xagawa, K. (2009). Efficient public key encryption based on ideal lattices. In *ASIACRYPT 2009*, volume 5479 of *LNCS*, pages 617–635, Cologne, Germany. springer.

A. Ideal Lattice-Based HVE Scheme

In this section we provide the HVE scheme using ideal lattice, detailing the security proof and showing how it is affected by this change.

A.1. Our Construction

Let n be the security parameter, σ be the Gaussian parameter, l be the length of vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, r the threshold value and $\mathcal{R} = \mathbb{Z}_q[x]/\langle x^n + 1\rangle$.

$\mathsf{Setup}(1^n)$. On input of security parameter n, the algorithm generates the public and secret keys as follows:

(i) run the $\mathsf{IdealTrapGen}(n, k, q, f, \sigma)$ algorithm to select uniformly $2l$ vectors $\hat{\boldsymbol{a}}_{i,b} \in \mathcal{R}^k$ (for $i \in [l]$ and $b \in \{0, 1\}$), with a short basis $\boldsymbol{T}_{\hat{\boldsymbol{a}}_{i,b}} \in \mathbb{Z}^{kn \times kn}$ for $\Lambda_q^{\perp}(\boldsymbol{A}_{i,b})$, such that $\boldsymbol{A}_{i,b} = \mathsf{Rot}_f(\hat{\boldsymbol{a}}_{i,b})$;
(ii) choose uniformly random vector $\boldsymbol{u} \in \mathcal{R}$;
(iii) output $mpk = (\{\hat{\boldsymbol{a}}_{i,b}\}, \boldsymbol{u})$ and $msk = \{\boldsymbol{T}_{\hat{\boldsymbol{a}}_{i,b}}\}$.

$\mathsf{KeyGen}(\boldsymbol{mpk}, \boldsymbol{msk}, \boldsymbol{v})$. On input of public-key mpk, master key msk and vector $\boldsymbol{v}$, the algorithm generates a secret key sk as follows:

(i) choose n random polynomials $p_i(x)$ of degree $r - 1$, with $p_i(0) = u_i$;
(ii) create l shares of vector $\boldsymbol{u}$ such that $\boldsymbol{u}_i = [p_1(i), ..., p_n(i)]$;
(iii) sample vectors for lattice $\Lambda(\boldsymbol{A}_{i,v_i})_q^{\boldsymbol{u}_i}$, with $\boldsymbol{A}_{i,v_i} = \mathsf{Rot}_f(\hat{\boldsymbol{a}}_{i,v_i})$ by invoking $\boldsymbol{e}_i \leftarrow \mathsf{SamplePre}(\boldsymbol{A}_{i,v_i}, \boldsymbol{T}_{\hat{\boldsymbol{a}}_{i,v_i}}, \boldsymbol{u}_i, \sigma) \in \mathbb{Z}^{kn}$ (for all i where $v_i = \star$, choose random $b \in \{0, 1\}$ and use $\hat{\boldsymbol{a}}_{i,b}$);
(iv) output $sk = \{\boldsymbol{e}_i\}$.

$\mathsf{Enc}(\boldsymbol{mpk}, \mathfrak{m}, \boldsymbol{w})$. On input of master public-key mpk, vector $\boldsymbol{w}$, and message $\mathfrak{m} \in \{0, 1\}$, the algorithm generates a ciphertext C as follows:

(i) choose a uniformly random vector $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$ and let $\beta = (l!)^2$;
(ii) choose a noise vector $\hat{\boldsymbol{x}}_i \leftarrow \overline{\Psi}_{\alpha_t}^{kn}$ and a noise term $x \leftarrow \overline{\Psi}_{\alpha_t}$;

(iii) compute $\boldsymbol{c}_i = \mathsf{Rot}_f(\hat{\boldsymbol{a}}_{i,w_i})^\top \boldsymbol{s} + \beta\hat{\boldsymbol{x}}_i \in \mathbb{Z}_q^{kn}$ and $c' = \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q$;
(iv) output $C = (\{\boldsymbol{c}_i\}, c')$.

$\mathsf{Dec}(mpk, sk, C)$. On input of master public key mpk, secret key sk, and ciphertext C, the algorithm does the following:

(i) let $\mathbb{J}$ be the set of matching bits between vectors $\boldsymbol{v}$ and $\boldsymbol{w}$, if $|\mathbb{J}| \geq r$; calculate each polynomial $l_j(x) = \prod_{i \in \mathbb{J}} \frac{x-i}{j-i}$;
(ii) the fractional Lagrangian coefficients will be $l_j = l_j(0)$ for $j \in \mathbb{J}$ so that $\sum l_j \boldsymbol{A}_{j,w_j} \boldsymbol{e}_j = \boldsymbol{u} \pmod q$;
(iii) compute $z = c' - \sum l_j \boldsymbol{e}_j^\top \boldsymbol{c}_j \pmod q$; for $z \in (-q/2, q/2]$, output 0 if $|z| < q/4$ and 1 otherwise.

A.2. Correctness

If $v_i = w_i$, for all $v_i \neq \star$, we have:

$$
\begin{aligned}
z &= c' - \sum l_j \boldsymbol{e}_j^\top \boldsymbol{c}_j \pmod q \\
z &= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \sum l_j \boldsymbol{e}_j^\top (\mathsf{Rot}_f(\hat{\boldsymbol{a}}_{j,w_j})^\top \boldsymbol{s} + \beta\hat{\boldsymbol{x}}_j) \pmod q \\
z &= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \sum l_j (\mathsf{Rot}_f(\hat{\boldsymbol{a}}_{j,w_j})\boldsymbol{e}_j)^\top \boldsymbol{s} - \sum l_j \boldsymbol{e}_j^\top \beta\hat{\boldsymbol{x}}_j \pmod q \\
z &= \boldsymbol{u}^\top \boldsymbol{s} + \beta x + \mathfrak{m} \cdot \lfloor q/2 \rceil - \boldsymbol{u}^\top \boldsymbol{s} - \sum l_j \boldsymbol{e}_j^\top \beta\hat{\boldsymbol{x}}_j \pmod q \\
z &= \mathfrak{m} \cdot \lfloor q/2 \rceil + \underbrace{\beta x - \sum l_j \boldsymbol{e}_j^\top \beta\hat{\boldsymbol{x}}_j}_{\text{error term}} \pmod q
\end{aligned}
$$

We have from Lemma 1 that $|\beta l_j| \leq \beta^2$, so we need to set the parameters in a way to guarantee that

$$\beta|x| + \sum \beta^2 |\boldsymbol{e}_j^\top \hat{\boldsymbol{x}}_j| < q/4$$

A.3. Security Reduction

In this section we prove the following theorem.

Theorem 10. *If the decision-*$\mathsf{ideal\text{-}LWE}_{q,n,\chi}$ *and decision-*$\mathsf{LWE}_{q,n,\chi}$ *problems are infeasible, then the HVE scheme described on Section A.1 is IND-sAT-CPA .*

Proof.

Init. $\mathcal{B}$ is given $\mathsf{ideal\text{-}LWE}$ l challenge pairs $(\hat{\boldsymbol{y}}_i, \hat{\boldsymbol{z}}_i) \in \mathcal{R}^k \times \mathcal{R}^k$ and a LWE pair $(\boldsymbol{v}, w) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$ where, either $\hat{\boldsymbol{z}}_i = \mathsf{Rot}_f(\hat{\boldsymbol{y}}_i)^\top \boldsymbol{s} + \beta\hat{\boldsymbol{x}}_i$ and $z = \langle \boldsymbol{y}, \boldsymbol{s} \rangle + \beta x$ for a random $\boldsymbol{s} \in \mathbb{Z}_q^n$, and noise terms $x \leftarrow \overline{\Psi}_{\alpha_t}$ and $\hat{\boldsymbol{x}}_i \leftarrow \overline{\Psi}_{\alpha_t}^{kn}$, or $\hat{\boldsymbol{z}}_i$ and z are uniformly random.

Setup. The public key is constructed using the vectors of the challenge pairs, as follows: $\hat{\boldsymbol{a}}_{i,w_i}$ will be $\beta\hat{\boldsymbol{y}}_i$, $\hat{\boldsymbol{a}}_{i,\overline{w_i}}$ will be computed using $\mathsf{IdealTrapGen}$ and $\boldsymbol{u}$ will be $\boldsymbol{y}$.

Secret keys. All private-key extraction queries are answered using the following algorithm: (i) let $\mathbb{J}$ be the set of matching bits between vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ and $|\mathbb{J}| < r$; (ii) for all $j \in \mathbb{J}$, use algorithm $\mathsf{SampleGaussian}(\mathsf{Rot}_f(\hat{\boldsymbol{a}}_{j,w_j}), \sigma)$ to find vector $\boldsymbol{e}_j$, such that $\boldsymbol{u}_j = \mathsf{Rot}_f(\hat{\boldsymbol{a}}_{j,w_j})\boldsymbol{e}_j$; (iii) choose randomly $r - 1 - |\mathbb{J}|$ vectors $\boldsymbol{u}_i$; (iii) represent each

vector $\boldsymbol{u}_i$ as $\boldsymbol{u}_i = \boldsymbol{u} + \boldsymbol{a}_1 i + ... + \boldsymbol{a}_{r-1} i^{r-1}$, since we already calculated r vectors, it is possible to compute all vectors $\boldsymbol{u}_i$, for all $i \in [l]$. (iv) find the remaining vector $\boldsymbol{e}_i$ by calling the $\mathsf{SamplePre}(\mathsf{Rot}_f(\hat{\boldsymbol{a}}_{i,w_i}), \boldsymbol{T}_{\hat{\boldsymbol{a}}_{i,w_i}}, \boldsymbol{u}_i, \sigma)$ algorithm.

Note that the distribution of the public parameters and keys in the real scheme is statistically indistinguishable from that in the simulation.

Challenge Ciphertext. The ciphertext $C = (\{\boldsymbol{c}_i\}, c')$ is constructed based on the terms on the ideal-LWE challenges pairs, $c' = \beta z + \mathfrak{m}\lfloor q/2 \rceil$ and $\boldsymbol{c}_i = \beta \hat{\boldsymbol{z}}_i$. If $\hat{\boldsymbol{z}}_i = \mathsf{Rot}_f(\hat{\boldsymbol{y}}_i)^\top \boldsymbol{s} + \hat{\boldsymbol{x}}_i$ on the ideal-LWE challenge and $z = \langle \boldsymbol{u}, \boldsymbol{s} \rangle + x$ on the LWE challenge, then the ciphertext is genuine; if $\hat{\boldsymbol{z}}_i$ and z are uniformly random, then the ciphertext is randomly generated.

Guess. $\mathcal{A}$ must guess whether it is interacting with a genuine or with a randomly generated ciphertext. The answer to this guess is also the answer to one of the LWE challenges. We showed that if z and all $\hat{\boldsymbol{z}}_i$ are uniformly random, then the ciphertext is randomly generated and if $\hat{\boldsymbol{z}}_i = \mathsf{Rot}_f(\hat{\boldsymbol{y}}_i)^\top \boldsymbol{s} + \beta \hat{\boldsymbol{x}}_i$ and $z = \langle \boldsymbol{y}, \boldsymbol{s} \rangle + \beta x$, then the ciphertext is genuine. Therefore, $\mathcal{B}$'s advantage in solving ideal-LWE is the same as $\mathcal{A}$'s advantage in distinguishing whether the ciphertext is genuine or not.

□