Lattice-Based Hierarchical Inner Product Encryption

Michel Abdalla[1], Angelo De Caro[2], and Karina Mochetti[3]

[1] Département d'Informatique, École Normale Supérieure, France
Michel.Abdalla@ens.fr
http://www.di.ens.fr/users/mabdalla

[2] Dipartimento di Informatica ed Applicazioni, Università degli Studi di Salerno, Italy
decaro@dia.unisa.it
http://www.dia.unisa.it/dottorandi/decaro/

[3] Instituto de Computação, UNICAMP, Brazil
mochetti@ic.unicamp.br
http://www.ic.unicamp.br/~mochetti/

Abstract. The notion of inner-product encryption (IPE), introduced by Katz, Sahai, and Waters at Eurocrypt 2008, is a generalization of identity-based encryption in which ciphertexts and secret keys are associated to vectors in some finite field. In an IPE scheme, a ciphertext can only be decrypted by a secret key if the vector associated with the latter is orthogonal to that of the ciphertext. In its hierarchical version, first proposed by Okamoto and Takashima (Asiacrypt'09), there exists an additional delegation mechanism which allows users to delegate their decryption capabilities to other users in the system. In this paper, we propose the first construction of a hierarchical inner-product encryption (HIPE) scheme based on lattices assumptions. To achieve this goal, we extend the lattice-based IPE scheme by Agrawal, Freeman, and Vaikuntanathan (Asiacrypt'11) to the hierarchical setting by employing basis delegation technics by Peikert et al. (Eurocrypt'10) and by Agrawal et al. (Eurocrypt'10). As the underlying IPE scheme, our new scheme is shown to be weak selective secure based on the difficulty of the learning with errors (LWE) problem in the standard model, as long as the total number of levels in the hierarchy is a constant. As an application, we show how our new primitive can be used to build new chosen-ciphertext secure IPE and wildcarded identity-based encryption schemes.

Keywords. Lattice-based cryptography, inner product, functional cryptography, hierarchical.

1 Introduction

Functional encryption has become quite popular in the last few years because it provides the system administrator with a fine-grained control over the decryption capabilities of its users. Such ability makes functional encryption schemes appealing in many emerging applications such as cloud services where the notion of public-key encryption reveals its inadequacy.

Even though several different flavors of functional encryption have appeared in the literature (see [7,9,13]), it was only recently that a systematic study of this notion has been proposed by Boneh, Sahai and Waters [8]. In their notion, each functional encryption scheme is associated with a functionality, which defines the set of admissible functions of the plaintext to which a secret decryption key may be linked. More specifically, in a functional encryption system for functionality $F(\cdot,\cdot)$, an authority holding a master secret key msk, can generate a key d_k that enables the computation of the function $F(k,\cdot)$ on the encrypted data. Then, using d_k the decryptor can compute $F(k,x)$ from an encryption of x. Important examples of functional encryption are attribute-based encryption (ABE) [20,12] and predicate encryption (PE) [9,13].

In this paper, we focus on the notion of inner-product encryption (IPE), introduced by Katz, Sahai, and Waters at Eurocrypt 2008 [13], which is a generalization of identity-based encryption. In an IPE scheme, ciphertexts and secret keys are associated to vectors in some finite field and a ciphertext can only be decrypted by a secret key if the vector associated with the latter is orthogonal to that of the ciphertext. In its hierarchical version (HIPE), first proposed by Okamoto and Takashima [16], there exists an additional delegation mechanism which allows users to delegate their decryption capabilities to other users in the system. As pointed out in [13], IPE is a very powerful primitive as it can be used to support conjunction, subset and range queries on encrypted data as well as disjunctions, polynomial evaluation, and CNF and DNF formulas. Moreover, the delegation mechanism proposed by Okamoto and Takashima [16] extends even further the capabilities of this primitive.

Since its introduction, there has been an extensive amount of work on the construction of IPE and HIPE schemes, most of them based on bilinear groups. The first IPE scheme was proposed by Katz et al. [13]. They proved the security of their scheme in the selective model, where the adversary must commit to the input on which it wishes to be challenged before seeing the public parameters, under variants of the subgroup decisional assumption in composite order bilinear groups. Later on, Okamoto et al. [16] were able to move to prime order bilinear groups introducing at the same time the concept of HIPE primitive. There the security was still proved in the selective model. More recently, Okamoto et al. [17] presented a HIPE scheme that achieves full security under the standard d-linear assumption on prime order bilinear groups.

In the lattice-based setting, the only known construction of an inner product scheme is due to Agrawal et al. [3], which was shown to be weak selective secure based on the difficulty of the learning with errors (LWE) problem. Informally, in a weak selective secure IPE scheme, the set of secret keys to which the adversary can query is more restricted than the one allowed in a standard selective secure definition.

Our Contributions. In this paper, we propose the first construction of a hierarchical inner-product encryption scheme based on lattices assumptions. To achieve this goal, we extend the lattice-based IPE scheme by Agrawal et al. [3] to the hierarchical setting by employing basis delegation technics by Peikert et al. [10] and by Agrawal et al. [2]. As the underlying IPE scheme, our new scheme is shown to be weak selective secure based on the difficulty of the learning with errors (LWE) problem in the standard model, as long as the total number of levels in the hierarchy is a constant. As an application, we show how our new primitive can be used to build new chosen-ciphertext secure IPE and wildcarded identity-based encryption schemes based on lattices.

2 Definitions

In this section we introduce hierarchical inner-product encryption primitive and the tools that we will use to implement it. In doing so, we adopt the same notation and definition style used in [16,14,3].

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 bold capital letters (e.g. $\boldsymbol{A}$) to denote matrices, bold lowercase letters (e.g. $\boldsymbol{w}$) to denote vectors that are components of our encryption scheme. The notation $\boldsymbol{A}^\top$ denotes the transpose of the 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 $\boldsymbol{A}_1$ 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. 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 $f(n)$ 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 $1 - \mathrm{negl}(n)$. The function $\lg x$ is the base 2 logarithm of x. The notation $\lfloor x \rceil$ denotes the nearest integer to x, rounding towards 0 for half-integers. The norm of a matrix $\boldsymbol{R} \in \mathbb{R}^{k\times m}$ is defined as $\|\boldsymbol{R}\| := \sup_{\|\boldsymbol{u}\|=1} \|\boldsymbol{R}\boldsymbol{u}\|$. The notation $[d]$ denotes the set of positive integers $\{1, 2, \ldots, d\}$.

2.1 Hierarchical Inner-Product Encryption

Let $\boldsymbol{\mu}$ be a tuple of positive integers $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$ such that $\sum_{i\in[d]} \mu_i = \ell$. We call $\boldsymbol{\mu}$ an *hierarchical format* of depth d. Let $\Sigma_{|h} = (\Sigma_1 \times \ldots \times \Sigma_h)$ where $h \leq d$ and $\Sigma_i = \mathbb{F}_N^{\mu_i}$ for finite field $\mathbb{F}_N$ of order N. A *hierarchical predicate* $f_{\boldsymbol{v}}$, with $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_h) \in \Sigma_{|h}$, is defined as follows: $f_{\boldsymbol{v}}(\boldsymbol{w}) = 1$, with $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t) \in \Sigma_{|t}$, if and only if $h \leq t$ and for all $i \in [h]$ we have that $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0$. An *hierarchical inner-product encryption* with hierarchical format $\boldsymbol{\mu}$ is defined by the following algorithms:

Setup$(1^\lambda, \boldsymbol{\mu})$. Takes as input security parameter λ and hierarchical format $\boldsymbol{\mu}$ and outputs public parameters mpk and master secret key msk.

Derive$(mpk, d_{\boldsymbol{v}}, \boldsymbol{v}_t)$. Takes as input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1}) \in \Sigma_{|t-1}$, and a vector $\boldsymbol{v}_t \in \Sigma_t$, and outputs a secret key $d_{\boldsymbol{v}'}$ for the vector $\boldsymbol{v}' = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1}, \boldsymbol{v}_t)$.

Enc$(mpk, \mathfrak{m}, \boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t) \in \Sigma_{|t})$. Takes as input public parameters $\mathfrak{m}$ in some associated message space, public parameters mpk and an attribute vector $\boldsymbol{w}$ and outputs a ciphertext C.

Dec$(mpk, C, d_{\boldsymbol{v}})$. Takes as input public parameters mpk, ciphertext C and secret key $d_{\boldsymbol{v}}$ and outputs the message $\mathfrak{m}$. We make the following consistency requirement. Suppose ciphertext C is obtained by running Enc on input mpk, message $\mathfrak{m}$ and attribute vector $\boldsymbol{w}$ and that $d_{\boldsymbol{v}}$ is a secret key for attribute vector $\boldsymbol{v}$ obtained through a sequence of Derive calls using the same mpk. Then Dec, on input mpk, C and $d_{\boldsymbol{v}}$, returns $\mathfrak{m}$, except with negligible probability, if and only if $f_{\boldsymbol{v}}(\boldsymbol{w}) = 1$.

Security Definition. Security is modeled by means of a game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$. In this work, we achieve *selective attribute* security, meaning that $\mathcal{A}$ must declare its *challenge attribute vectors* $(\boldsymbol{w}_0, \boldsymbol{w}_1)$ before seeing the public key. Moreover, $\mathcal{A}$ is allowed to ask queries before and even after seeing the challenge ciphertext but it is required $\mathcal{A}$ to ask for keys of predicates $\boldsymbol{v}$ that cannot decrypt the challenge ciphertext; that is, for which $f_{\boldsymbol{v}}(\boldsymbol{w}_0) = f_{\boldsymbol{v}}(\boldsymbol{w}_1) = 0$. This notion is called *weak attribute hiding*. Specifically, the game is defined in the following way:

Init. $\mathcal{A}$ is given hierarchical format $\boldsymbol{\mu}$ of depth d and outputs challenge vectors $\boldsymbol{w}_0$, $\boldsymbol{w}_1 \in \Sigma_{|h}$.

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

Phase 1. $\mathcal{A}$ is given oracle access to Derive$(mpk, msk, \cdot)$. Then, $\mathcal{A}$ can delegate secret keys directly by invoking the Derive algorithm.

Challenge. $\mathcal{A}$ gives a pair of message $(\mathfrak{m}_0, \mathfrak{m}_1)$ to $\mathcal{C}$. We require that for all attribute vectors $\boldsymbol{v}$ derivable from any revealed secret key in Phase 1, $f_{\boldsymbol{v}}(\boldsymbol{w}_0) = f_{\boldsymbol{v}}(\boldsymbol{w}_1) = 0$. Then $\mathcal{C}$ chooses random $\eta \overset{\$}{\leftarrow} \{0,1\}$, encrypts $\mathfrak{m}_\eta$ under $\boldsymbol{w}_\eta$ and sends the resulting ciphertext to $\mathcal{A}$.

Phase 2. The same as Phase 1 with the same restriction of the Challenge phase.

Guess. $\mathcal{A}$ must output a guess β' for β. The advantage of $\mathcal{A}$ is defined to be $\Pr[\eta' = \eta] - \frac{1}{2}$.

Definition 1. An hierarchical inner-product encryption scheme is *weak attribute hiding-selective attribute* secure (IND-wAH-sAT-HIPE-CPA for short), if all polynomial time adversaries achieve at most a negligible (in λ) advantage in the previous security game. ▮

2.2 Lattices

In this section we collect results from [2,3,5,15,11,10] that we will need for our construction and the 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

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

The lattice $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A})$ is a coset of $\Lambda_q^\perp(\boldsymbol{A})$; namely, $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}) = \Lambda_q^\perp(\boldsymbol{A}) + \boldsymbol{t}$ for any $\boldsymbol{t}$ such that $\boldsymbol{A}\cdot\boldsymbol{t} = \boldsymbol{u} \mod q$.

Gram-Schmidt Norm. 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., $\max_{1 \le i \le k} \|\boldsymbol{s}_i\|$, 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 $\boldsymbol{S}$.

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 $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{w}) := \exp\left(-\pi\|x - \boldsymbol{c}\|^2/\sigma^2\right)$ 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 lemma captures standard properties of these distributions.

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

1. $\Pr\left[\|\boldsymbol{w} - \boldsymbol{c}\| > \sigma\sqrt{m} \ : \ \boldsymbol{w} \xleftarrow{\$} \mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}\right] \le \mathrm{negl}(n)$
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* SampleGaussian$(\boldsymbol{A}, \boldsymbol{T}_{\boldsymbol{A}}, \sigma, \boldsymbol{c})$ *that returns $\boldsymbol{x} \in \Lambda_q^{\perp}(\boldsymbol{A})$ drawn from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}$.*
4. *There is a PPT algorithm* SamplePre$(\boldsymbol{A}, \boldsymbol{T}_{\boldsymbol{A}}, \boldsymbol{u}, \sigma)$ *that returns $\boldsymbol{x} \in \Lambda_q^{\perp}(A)$ sampled from a distribution statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{A}),\sigma}$, whenever $\Lambda_q^{\boldsymbol{u}}(A)$ is not empty.*

∎

The Norm of a Random Matrix. The following lemmata can be used to bound the norm of a random matrix in $\{-1, 1\}^{m \times m}$.

Lemma 3. *([2, Lemma 15]) Let $\boldsymbol{R}$ be a $k \times m$ matrix chosen at random from $\{-1, 1\}^{k \times m}$. Then* $\Pr\left[\|\boldsymbol{R}\| > 12\sqrt{k + m}\right] < e^{-(k+m)}$. ∎

Lemma 4. *([2, Lemma 16]) Let $\boldsymbol{u} \in \mathbb{R}^m$ be some vector of norm 1. Let $\boldsymbol{R}$ be a $k \times m$ matrix chosen at random from $\{-1, 1\}^{k \times m}$. Then* $\Pr[\|\boldsymbol{R}\boldsymbol{u}\| > \sqrt{k}\omega(\sqrt{\log k})] < \mathrm{negl}(k)$. ∎

Sampling Algorithms. Following [2,10,4,5] we will need the following algorithms to sample short vectors and random basis from specific lattices.

Algorithm ToBasis. Micciancio and Goldwassser [31] showed that a full-rank set $\boldsymbol{S}$ in a lattice Λ can be converted into a basis T for Λ with an equally low Gram-Schmidt norm.

Lemma 5. *([31, Lemma 7.1]) Let Λ be an m-dimensional lattice. There is a deterministic polynomial-time algorithm that, given an arbitrary basis of Λ and a full-rank set $\boldsymbol{S} = \boldsymbol{s}_1, \ldots, \boldsymbol{s}_m$ in Λ, returns a basis T of Λ satisfying $\|\widetilde{\boldsymbol{T}}\| \le \|\widetilde{\boldsymbol{S}}\|$ and $\|\boldsymbol{T}\| \le \|\boldsymbol{S}\|\sqrt{m}/2$* ∎

Algorithm TrapGen. Ajtai [4] and later Alwen and Peikert [5] showed 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 6. *([5, 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 $(\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}, \boldsymbol{S} \in \mathbb{Z}^{m\times m})$ 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 $\|\boldsymbol{S}\| \leq O(n \log q)$ w.o.p. in n.* ∎

We let $\sigma_{\mathrm{TG}} = O(\sqrt{n \log q})$ denote the maximum with high probability Gram-Schmidt norm of a basis produced by $\mathsf{TrapGen}$.

Algorithm ExtendBasis. Peikert et al. [10] shows how to construct a basis for $\Lambda_q^{\perp}(\boldsymbol{A}\|\boldsymbol{B}\|\boldsymbol{C})$ from a basis for $\Lambda_q^{\perp}(\boldsymbol{B})$.

Theorem 7. *For $i = 1, 2, 3$ let $\boldsymbol{A}_i$ be a matrix in $\mathbb{Z}_q^{n\times m_i}$ and let $\boldsymbol{A} := (\boldsymbol{A}_1\|\boldsymbol{A}_2\|\boldsymbol{A}_3)$. Let $\boldsymbol{T}_2$ be a basis of $\Lambda_q^{\perp}(\boldsymbol{A}_2)$. There is deterministic polynomial time algorithm* $\mathsf{ExtendBasis}(\boldsymbol{A}_1, \boldsymbol{A}_2, \boldsymbol{A}_3, \boldsymbol{T}_2)$ *that outputs a basis T for $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\widetilde{\boldsymbol{T}}\| = \|\widetilde{\boldsymbol{T}_2}\|$* ∎

Algorithm SampleLeft. The algorithm takes as input a full rank matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, a short basis $\boldsymbol{T_A}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$, a matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m_1}$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter σ. Let $\boldsymbol{F} := (\boldsymbol{A}\|\boldsymbol{B})$, then the algorithm outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ in the coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F})$.

Theorem 8. *([2, Theorem 17], [10, Lemma 3.2]) Let $q > 2, m > n$ and $\sigma > \|\boldsymbol{T_A}\| \cdot \omega(\sqrt{\log(m + m_1)})$. Then* $\mathsf{SampleLeft}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{T_A}, \boldsymbol{u}, \sigma)$ *outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{m+m_1}$ statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}),\sigma}$.* ∎

Algorithm SampleRight. The algorithm takes as input matrices $\boldsymbol{A} \in \mathbb{Z}_q^{n\times k}$ and $\boldsymbol{R} \in \mathbb{Z}^{k\times m}$, a full rank matrix $\boldsymbol{B} \in \mathbb{Z}_q^{n\times m}$ and a short basis $\boldsymbol{T_B}$ of $\Lambda_q^{\perp}(\boldsymbol{B})$, a vector $\boldsymbol{u} \in \mathbb{Z}_q^n$, and a Gaussian parameter σ. Let $\boldsymbol{F} := (\boldsymbol{A}\|\boldsymbol{AR} + \boldsymbol{B})$, then the algorithm outputs a vector $\boldsymbol{e} \in \mathbb{Z}^{k+m}$ in the coset $\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F})$.

Often the matrix $\boldsymbol{R}$ given to the algorithm as input will be a random matrix in $\{-1, 1\}^{m\times m}$. Let S^m be the m-sphere $\{\boldsymbol{x} \in \mathbb{R}^{m+1} : \|\boldsymbol{x}\| = 1\}$. We define $s_{\boldsymbol{R}} := \|\boldsymbol{R}\| = \sup_{\boldsymbol{x}\in S^{m-1}} \|\boldsymbol{R} \cdot \boldsymbol{x}\|$.

Theorem 9. *([2, Theorem 19]) Let $q > 2, m > n$ and $\sigma > \|\boldsymbol{T_B}\| \cdot s_R \cdot \omega(\sqrt{\log(k + m)})$. Then* $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{R}, \boldsymbol{T_B}, \boldsymbol{u}, \sigma)$ *outputs a vector $\boldsymbol{e} \in Z^{k+m}$ distributed statistically close to $\mathcal{D}_{\Lambda_q^{\boldsymbol{u}}(\boldsymbol{F}),\sigma}$.* ∎

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 10. ([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 $(\boldsymbol{a}, \langle\boldsymbol{a}, \boldsymbol{r}\rangle + e)$.

(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 $\mathbb{Z}_q^n \times Z_q$, output 0 if the former holds and 1 if the latter holds.

We say the *decision*-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible if for all polynomial-time algorithms $\mathcal{A}$, the probability that $\mathcal{A}$ solves the *decision*-LWE problem (over $\boldsymbol{r}$ and $\mathcal{A}$s random coins) is negligibly close to $1/2$ as a function of n. ▮

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

Definition 11. 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 12. *([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 GapSVP to within $\widetilde{O}(n/\alpha)$ in the worst case.* ▮

Theorem 13. *([18]) Let n, q be integers and $\alpha \in (0,1)$, and $q = \sum_i q_i \leq 2^{n/2}$, where the q_i are distinct primes satisfying $\omega(\log n)/\alpha \leq q_i \leq \mathrm{poly}(n)$. If there exists an efficient (classical) algorithm that solves decision-$\mathsf{LWE}_{q,n,\overline{\Psi}_\alpha}$, then there exists an efficient (classical) algorithm that approximates GapSVP to within $\widetilde{O}(n/\alpha)$ in the worst case.* ▮

The following lemma will be used to show correctness of decryption.

Lemma 14. *([2, 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 $|\langle \boldsymbol{e}, \boldsymbol{v}\rangle| \leq \|\boldsymbol{e}\| \cdot \big(q\alpha \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2\big)$ w.o.p. (in m).* ▮

3 Hierarchical Inner Product Encryption Scheme

This section describes our hierarchical inner-product encryption scheme based on [3].

Intuitions. For hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$, the public parameters will contain random matrices $(\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\})$ in $\mathbb{Z}_q^{n\times m}$. The master secret key is a trapdoor $\boldsymbol{T_A}$ for $\boldsymbol{A}$. To generate a secret key for vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ at depth $t \leq d$ we use the matrix:

$$F_{\boldsymbol{v}} = \left(\boldsymbol{A}\| \sum_{j\in[\mu_1]} \sum_{\gamma=0}^{k} v_{1,j,\gamma} \cdot \boldsymbol{A}_{1,j,\gamma}\| \ldots \| \sum_{j\in[\mu_t]} \sum_{\gamma=0}^{k} v_{t,j,\gamma} \cdot \boldsymbol{A}_{t,j,\gamma}\right) \in \mathbb{Z}_q^{n\times(t+1)m} \tag{1}$$

where each $v_{i,j}$ is r-decomposed for a certain fixed r and $k = \lfloor \log_r q \rfloor$. Then the secret key for $\boldsymbol{v}$ is a short basis for the lattice $\Lambda_q^\perp(F_{\boldsymbol{v}})$. By using the short basis for $\Lambda_q^\perp(F_{\boldsymbol{v}})$ is possible to generate a random short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}\|\boldsymbol{v}_{t+1}})$. This provides the delegation mechanism.

Let us sketch briefly the security reduction. For challenge vector $\boldsymbol{w}^\star = (\boldsymbol{w}_1^\star, \ldots, \boldsymbol{w}_{t^\star}^\star)$, the simulator chooses the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ uniformly at random in $\mathbb{Z}_q^{n\times m}$ and construct the matrices $\boldsymbol{A}_{i,j,\gamma}$ as follows:

$$\boldsymbol{A}_{i,j,\gamma} = \boldsymbol{A}\boldsymbol{R}_{i,j,\gamma} - r^{\gamma} w_{i,j}^\star \boldsymbol{B}$$

where $\boldsymbol{R}_{i,j,\gamma} \in \{-1,1\}^{m\times m}$. Since the matrices $\boldsymbol{A}, \{\boldsymbol{R}_{i,j,\gamma}\}$ are uniform and independent in $\mathbb{Z}_q^{n\times m}$, we have that the $\boldsymbol{A}_{i,j,\gamma}$'s are uniform in $\mathbb{Z}_q^{n\times m}$ as in the real system. Moreover the simulator has a trapdoor $\boldsymbol{T_B}$ for $\Lambda_q^{\perp}(\boldsymbol{B})$ but no trapdoor for $\Lambda_q^{\perp}(\boldsymbol{A})$. To generate a secret key for vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$, the simulator must produce a short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ where

$$F_{\boldsymbol{v}} = \left(\boldsymbol{A} \| \boldsymbol{A}\overline{\boldsymbol{R}_1} - \langle \boldsymbol{v}_1, \boldsymbol{w}_1^\star\rangle \boldsymbol{B} \| \ldots \| \boldsymbol{A}\overline{\boldsymbol{R}_t} - \langle \boldsymbol{v}_t, \boldsymbol{w}_t^\star\rangle \boldsymbol{B}\right)$$

Then let

$$\begin{aligned}
\overline{\boldsymbol{R}_i} &= \sum_{j\in[\mu_i]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \boldsymbol{R}_{i,j,\gamma} \quad &&\in \mathbb{Z}_q^{m\times m} \\
\overline{\boldsymbol{R}} &= \left[\overline{\boldsymbol{R}_1} \| \ldots \| \overline{\boldsymbol{R}_t}\right] \quad &&\in \mathbb{Z}_q^{m\times t\cdot m} \\
\boldsymbol{B_v} &= [-\langle \boldsymbol{v}_1, \boldsymbol{w}_1^\star\rangle \boldsymbol{B} \| \ldots \| - \langle \boldsymbol{v}_t, \boldsymbol{w}_t^\star\rangle \boldsymbol{B}] \quad &&\in \mathbb{Z}_q^{n\times t\cdot m}
\end{aligned}$$

Thus $F_{\boldsymbol{v}}$ can be written as:

$$F_{\boldsymbol{v}} = \left(\boldsymbol{A} \| \boldsymbol{A}\overline{\boldsymbol{R}} + \boldsymbol{B_v}\right) \quad \in \mathbb{Z}_q^{n\times(t+1)m} \,. \tag{2}$$

When $\boldsymbol{v}$ is not a prefix of $\boldsymbol{w}$ meaning that there exists an index i such that $\langle \boldsymbol{v}_i, \boldsymbol{w}_i^\star\rangle \neq 0$, the simulator can then extend $\boldsymbol{T_B}$ to a short basis for the entire lattice $\Lambda_q^{\perp}(\boldsymbol{B_v})$. The simulator can now generate short vectors in $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$ using algorithm SampleRight, which is sufficient for constructing a short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$, as required. When $\boldsymbol{v}$ is a prefix of $\boldsymbol{w}$, meaning that for each $i = 1, \ldots, t$, $\langle \boldsymbol{v}_i, \boldsymbol{w}_i^\star\rangle = 0$, then the matrix $F_{\boldsymbol{v}}$ no longer depends on $\boldsymbol{B}$ and the simulators trapdoor disappears. Consequently, the simulator can generate secret keys for all vectors other than prefixes of $\boldsymbol{w}^\star$. As we will see, for $\boldsymbol{w}^\star$ the simulator can produce a challenge ciphertext that helps it solve the given LWE challenge.

3.1 Sampling a Random Basis

In this Section we describe the algorithms that we will use to realize the delegation mechanism and for the simulation. Following [2,10], let Λ be an m-dimensional lattice and let $\mathcal{O}(\Lambda, \sigma)$ be an algorithm that generates independent samples from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma}$. The following algorithm called $\mathsf{SampleBasis}^{\mathcal{O}}(\Lambda, \sigma)$ uses $\mathcal{O}$ to generate a basis T of Λ in the following way:

1. For $i = 1, \ldots, m$, generate $v \overset{\$}{\leftarrow} \mathcal{O}(\Lambda, \sigma)$, if v is independent of $\{v_1, \ldots, v_{i-1}\}$, set $v_i \leftarrow v$, if not, repeat.
2. Convert the set of independent vectors $v_1, \ldots, v_m$ to a basis T using Lemma 6 (and using some canonical basis of Λ) and output T.

The following theorem summarizes properties of this algorithm.

Lemma 15. *For $\sigma > \tilde{bl}(\Lambda)\omega(\sqrt{\log m})$ algorithm* $\mathsf{SampleBasis}^{\mathcal{O}}(\Lambda, \sigma)$ *satisfies the following properties:*

1. *Step 1 requires at most $O(m \log m)$ w.h.p and $2m$ samples in expectation.*
2. *With overwhelming probability $\|\widetilde{T}\| \leq \|T\| \leq \sigma\sqrt{m}$.*
3. *Up to a statistical distance, the distribution of T does not depend on the implementation of $\mathcal{O}$. That is, the random variable* $\mathsf{SampleBasis}^{\mathcal{O}}(\Lambda, \sigma)$ *is statistically close to* $\mathsf{SampleBasis}^{\mathcal{O}'}(\Lambda, \sigma)$ *for any algorithm $\mathcal{O}'$ that samples from a distribution statistically close to $\mathcal{D}_{\Lambda,\sigma}$.*

∎

Algorithm SampleBasisLeft. We are interested in the lattice $\Lambda_q^\perp(F_{\boldsymbol{v}})$ where $F_{\boldsymbol{v}}$ is defined in (1) for $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$. Write $F_{\boldsymbol{v}} = (\boldsymbol{A}|\boldsymbol{M})$ for some matrices $\boldsymbol{A}$ and $\boldsymbol{M}$, then given a short basis $\boldsymbol{T_A}$ for $\Lambda_q^\perp(\boldsymbol{A})$ we can implement algorithm $\mathcal{O}(F_{\boldsymbol{v}}, \sigma)$ by invoking $\mathsf{SampleLeft}(\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{T_A}, 0, \sigma)$. When $\sigma > \|\widetilde{\boldsymbol{T_A}}\| \cdot \omega(\sqrt{\log((t+1)m)})$, Theorem 8 shows that the resulting vector is distributed statistically close to $\mathcal{D}_{\Lambda_q^\perp(F_{\boldsymbol{v}}),\sigma}$ as required for SampleBasis. Using the above algorithm in algorithm SampleBasis leads to an algorithm to sample a random basis of $\Lambda_q^\perp(F_{\boldsymbol{v}})$ given a short basis of $\boldsymbol{A}$. We refer to this algorithm as SampleBasisLeft and summarize its properties in the following corollary.

Corollary 16. Algorithm $\mathsf{SampleBasisLeft}(\boldsymbol{A}, \boldsymbol{M}, \boldsymbol{T_A}, \sigma)$ outputs a basis of $\Lambda_q^\perp(F_{\boldsymbol{v}})$ satisfying the three properties in Lemma 15 provided that $\boldsymbol{A}$ is rank n and $\sigma > \|\widetilde{\boldsymbol{T_A}}\| \cdot \omega\left(\sqrt{\log((t+1)m)}\right)$. ∎

Algorithm SampleBasisRight. In the simulation, the matrix $F_{\boldsymbol{v}}$ is defined as in (2). In this case, given a short basis $\boldsymbol{T_B}$ for $\Lambda_q^\perp(\boldsymbol{B})$ we can implement algorithm $\mathcal{O}(F_{\boldsymbol{v}}, \sigma)$ as follows:

1. Using Theorem 7, extend basis $\boldsymbol{T_B}$ for $\Lambda_q^\perp(\boldsymbol{B})$ to a basis $\boldsymbol{T_{B_v}}$ for $\Lambda_q^\perp(\boldsymbol{B_v})$ such $\|\widetilde{\boldsymbol{T_{B_v}}}\| = \|\widetilde{\boldsymbol{T_B}}\|$.
2. Then run $\mathsf{SampleRight}(\boldsymbol{A}, \boldsymbol{B_v}, \overline{\boldsymbol{R}}, \boldsymbol{T_{B_v}}, 0, \sigma)$ and output the result. When $\boldsymbol{B_v}$ is rank n and $\boldsymbol{v}$ is not a prefix of $\boldsymbol{w}^\star$ the matrix $\boldsymbol{B_v}$ is rank n as required for SampleRight.

Let $s_{\overline{\boldsymbol{R}}} := \|\overline{\boldsymbol{R}}\|$ be the norm of the matrix $\overline{\boldsymbol{R}}$. When $\sigma > \|\widetilde{\boldsymbol{T_B}}\| \cdot s_{\overline{\boldsymbol{R}}} \cdot \omega(\sqrt{\log((t+1)m)})$, Theorem 9 shows that the resulting vector is distributed statistically close to $\mathcal{D}_{\Lambda_q^\perp(F_{\boldsymbol{v}}),\sigma}$ as required for SampleBasis. Using the above algorithm in algorithm SampleBasis leads to an algorithm to sample a random basis of $\Lambda_q^\perp(F_{\boldsymbol{v}})$ for $F_{\boldsymbol{v}}$ defined in (2) given a short basis of $\boldsymbol{B}$. We refer to this algorithm as SampleBasisRight and summarize its properties in the following corollary.

Corollary 17. Algorithm $\mathsf{SampleBasisRight}(\boldsymbol{A}, \boldsymbol{B_v}, \overline{\boldsymbol{R}}, \boldsymbol{T_B}, \sigma)$ outputs a basis of $\mathcal{D}_{\Lambda_q^{\perp}(F_{\boldsymbol{v}}),\sigma}$ satisfying the three properties in Lemma 15 provided that $\boldsymbol{B}$ is rank n, that $\boldsymbol{v}$ is not a prefix of $\boldsymbol{w}^{\star}$ and $\sigma > \|\widetilde{\boldsymbol{T_B}}\| \cdot s_{\overline{\boldsymbol{R}}} \cdot \omega(\sqrt{\log((t+1)m)})$. ∎

3.2 Our Construction

Let $n > 0$ be the security parameter and $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$ the hierarchical format. Let $q = q(n, \boldsymbol{\mu})$ and $m = m(n, \boldsymbol{\mu})$ be positive integers. Let $r = r(n, \boldsymbol{\mu}) \geq 2$ be and integer and define $k = k(n, \boldsymbol{\mu}) := \lfloor \log_r q \rfloor$. Our *hierarchical inner-product encryption* for hierarchical format $\boldsymbol{\mu}$ consists of the following algorithms.

Setup$(1^n, \boldsymbol{\mu})$. On input a security parameter n, and an hierarchical format of depth d $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$, the algorithm generates public and secret parameters as follows: Use algorithm $\mathsf{TrapGen}(q, n, m)$ to select a uniformly random $n \times m$-matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ with a basis $\boldsymbol{T_A} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\widetilde{\boldsymbol{T_A}}\| \leq O(\sqrt{n \log q})$. For $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$, choose uniformly random matrices $\boldsymbol{A}_{i,j,\gamma} \in \mathbb{Z}_q^{n \times m}$. Select a uniformly random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. Output $mpk = (\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \boldsymbol{u})$ and $msk = \boldsymbol{T_A}$.

Derive$(mpk, d_{\boldsymbol{v}}, \boldsymbol{v_t})$. On input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{t-1})$, and the vector $\boldsymbol{v}_t$, the algorithm generates a secret key for the vector $\boldsymbol{v'} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ as follows: Construct short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v'}})$ by invoking $\boldsymbol{S} \leftarrow \mathsf{SampleBasisLeft}(F_{\boldsymbol{v}}, \sum_{j \in [\mu_i]} \sum_{\gamma \in k} v_{t,j,\gamma} \cdot \boldsymbol{A}_{t,j,\gamma}, d_{\boldsymbol{v}}, \sigma_t)$ where $F_{\boldsymbol{v'}} = \left[F_{\boldsymbol{v}} \| \sum_{j \in [\mu_i]} \sum_{\gamma=0}^{k} v_{t,j,\gamma} \cdot \boldsymbol{A}_{t,j,\gamma}\right]$ and $d_{\boldsymbol{v}}$ is a short basis for $\Lambda_q^{\perp}(F_{\boldsymbol{v}})$. By Corollary 16, when $\sigma_t > \|\widetilde{d_{\boldsymbol{v}}}\| \cdot \omega(\sqrt{\log((t+1)m)})$ then $\|\widetilde{d_{\boldsymbol{v'}}}\| \leq \|d_{\boldsymbol{v'}}\| \leq \sigma_t \cdot \sqrt{(t+1)m}$. Output $d_{\boldsymbol{v'}} = \boldsymbol{S}$. Notice that, for the special case of the first level secret keys when $\boldsymbol{v}$ is the empty vector ϵ, we define $F_\epsilon := \boldsymbol{A}$ and $d_{\boldsymbol{v}} = msk$.

Enc$(mpk, \boldsymbol{w}, \mathfrak{m})$. On input the master public key mpk, the vector $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$, and the message $\mathfrak{m} \in \{0, 1\}$, the algorithm generates a ciphertext C as follows: Choose a uniformly random matrix $\boldsymbol{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$. Choose a noise vector $\boldsymbol{x} \leftarrow \overline{\Psi}_{\alpha_t}^m$ and a noise term $x \leftarrow \overline{\Psi}_{\alpha_t}$. Compute $\boldsymbol{c}_0 = \boldsymbol{A}^{\top} \boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^m$. For $i \in [t]$, $j \in [\mu_i]$ and $\gamma = 0, \ldots, k$ choose a random matrix $\boldsymbol{R}_{i,j,\gamma} \in \{-1, 1\}^{m \times m}$ and compute $\boldsymbol{c}_{i,j,\gamma} = (\boldsymbol{A}_{i,j,\gamma} + r^{\gamma} w_{i,j} \boldsymbol{B})^{\top} \boldsymbol{s} + \boldsymbol{R}_{i,j,\gamma}^{\top} \boldsymbol{x} \in \mathbb{Z}_q^m$. Compute $c' = \boldsymbol{u}^{\top} \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q$. Output $C = (\boldsymbol{c}_0, \{\boldsymbol{c}_{i,j,\gamma}\}, c')$.

Dec$(mpk, d_{\boldsymbol{v}}, C)$. On input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$, and a ciphertext $C = (\boldsymbol{c}_0, \{\boldsymbol{c}_{i,j,\gamma}\}, c')$, the algorithm does the following: For $i \in [t]$ define the r-ary expansion of the vector $\boldsymbol{v}_i$ and compute $\boldsymbol{c}_{\boldsymbol{v}_i} = \sum_{j \in [\mu_i]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \boldsymbol{c}_{i,j,\gamma}$. Let $\boldsymbol{c} = [\boldsymbol{c}_0 \| \boldsymbol{c}_{\boldsymbol{v}_1} \| \ldots \| \boldsymbol{c}_{\boldsymbol{v}_t}]$. Set $\tau_t := \sigma_t \cdot \sqrt{(t+1)m} \cdot \omega(\sqrt{(t+1)m})$. Then $\tau_t \geq \|\widetilde{d_{\boldsymbol{v}}}\| \cdot \omega(\sqrt{(t+1)m})$. Compute $\boldsymbol{e_v} = \mathsf{SamplePre}(F_{\boldsymbol{v}}, d_{\boldsymbol{v}}, \boldsymbol{u}, \tau_t)$ and $z = c' - \boldsymbol{e}_{\boldsymbol{v}}^{\top} \cdot \boldsymbol{c}$.

Interpreter z as in integer in $(-q/2, q/2]$, then output 0 if $|z| < q/4$ and 1 otherwise.

3.3 Correctness

Lemma 18. *For hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$ of depth d, suppose the parameters q and α_t, for each $t \in [d]$, are such that $q/\log q = \Omega(\sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m^{3/2})$ and $\alpha_t \le (t \cdot \log q \cdot \sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m \cdot \omega(\sqrt{\log m}))^{-1}$, where $\mu = \max_{i \in [d]} \mu_i$. Moreover, for vectors $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ and $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_h)$, let $d_{\boldsymbol{v}}$ be the basis of the lattice $F_{\boldsymbol{v}}$ obtained through a sequence of calls to the* Derive *algorithm, let $C \leftarrow \mathsf{Enc}(mpk, \boldsymbol{w}, \mathfrak{m})$ and $\mathfrak{m}' \leftarrow \mathsf{Dec}(mpk, d_{\boldsymbol{v}}, C)$.*

Then, if $f_{\boldsymbol{v}}(\boldsymbol{w}) = 1$, namely $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0 \pmod q$ for each $i \in [t]$, then with overwhelming probability we have $\mathfrak{m}' = \mathfrak{m}$. ∎

Proof. Just for notation simplification, let

$$\widetilde{\boldsymbol{A}_i} = \left(\sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{A}_{i,j,\gamma} \right) \quad \text{and} \quad \widetilde{\boldsymbol{R}_i} = \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{R}_{i,j,\gamma}^{\top} \boldsymbol{x} \ .$$

Then, for each $i \in [t]$, the decryption algorithm computes:

$$\begin{aligned} \boldsymbol{c}_{\boldsymbol{v}_i} &= \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \boldsymbol{c}_{i,j,\gamma} \\ &= \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \cdot \left[(\boldsymbol{A}_{i,j,\gamma} + r^{\gamma} w_{i,j} \boldsymbol{B})^{\top} \boldsymbol{s} + \boldsymbol{R}_{i,j,\gamma}^{\top} \boldsymbol{x} \right] \\ &= \widetilde{\boldsymbol{A}_i}^{\top} \boldsymbol{s} + \underbrace{\left(\sum_{j \in [\mu]} \sum_{\gamma=0}^{k} r^{\gamma} v_{i,j,\gamma} w_{i,j} \right)}_{\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle} \boldsymbol{B}^{\top} \boldsymbol{s} + \widetilde{\boldsymbol{R}_i} \ . \end{aligned}$$

If $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0$ then we have:

$$\boldsymbol{c}_{\boldsymbol{v}_i} = \widetilde{\boldsymbol{A}_i}^{\top} \boldsymbol{s} + \widetilde{\boldsymbol{R}_i} \pmod q \ .$$

Thus, if for each $i \in [t]$, $\langle \boldsymbol{v}_i, \boldsymbol{w}_i \rangle = 0$ then $\boldsymbol{c}$ can be written as:

$$\begin{aligned} \boldsymbol{c} =& [\boldsymbol{c}_0 \| \boldsymbol{c}_{\boldsymbol{v}_1} \| \ldots \| \boldsymbol{c}_{\boldsymbol{v}_t}] \\ =& \left[\boldsymbol{A} \| \widetilde{\boldsymbol{A}_1} \| \ldots \| \widetilde{\boldsymbol{A}_t} \right]^{\top} \boldsymbol{s} + \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \ldots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod q \\ =& F_{\boldsymbol{v}}^{\top} \cdot \boldsymbol{s} + \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \ldots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod q \ . \end{aligned}$$

At this point, the short vector $\boldsymbol{e}_{\boldsymbol{v}} = \mathsf{SampleLeft}(F_{\boldsymbol{v}}, d_{\boldsymbol{v}}, \boldsymbol{u}, \tau_t)$ is computed by the decryption algorithm such that by Theorem 8, $F_{\boldsymbol{v}} \cdot \boldsymbol{e}_{\boldsymbol{v}} = \boldsymbol{u} \pmod q$. It follows that

$$\boldsymbol{e}_{\boldsymbol{v}}^{\top} \boldsymbol{c} = \boldsymbol{u}^{\top} \boldsymbol{s} + \boldsymbol{e}_{\boldsymbol{v}}^{\top} \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \ldots \| \widetilde{\boldsymbol{R}_t} \right]^{\top} \pmod q \ .$$

Finally, the decryption algorithm computes:

$$\begin{aligned} z &= c' - \boldsymbol{e}_{\boldsymbol{v}}^\top \boldsymbol{c} \pmod q \\ &= \left(\boldsymbol{u}^\top \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil\right) - \boldsymbol{u}^\top \boldsymbol{s} - \boldsymbol{e}_{\boldsymbol{v}}^\top \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \ldots \| \widetilde{\boldsymbol{R}_t}\right]^\top \pmod q \\ &= \mathfrak{m} \cdot \lfloor q/2 \rceil + \underbrace{\left(x - \boldsymbol{e}_{\boldsymbol{v}}^\top \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \ldots \| \widetilde{\boldsymbol{R}_t}\right]^\top\right)}_{\text{noise term}} \pmod q \,. \end{aligned}$$

Thus, to have a successful decryption, it suffices to set the parameters so that with overwhelming probability,

$$\left| x - \boldsymbol{e}_{\boldsymbol{v}}^\top \left[\boldsymbol{x} \| \widetilde{\boldsymbol{R}_1} \| \ldots \| \widetilde{\boldsymbol{R}_t}\right]^\top \right| < \frac{q}{4} \,.$$

Let us write $\boldsymbol{e}_{\boldsymbol{v}} = [\boldsymbol{e}_{\boldsymbol{v},0} \| \boldsymbol{e}_{\boldsymbol{v},1} \| \ldots \| \boldsymbol{e}_{\boldsymbol{v},t}]$ with $\boldsymbol{e}_{\boldsymbol{v},i} \in \mathbb{Z}^m$ for $i = 0, \ldots, t$. Then the noise term can be rewritten as

$$x - \left(\boldsymbol{e}_{\boldsymbol{v},0} + \sum_{i \in [t]} \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{R}_{i,j,\gamma} \boldsymbol{e}_{\boldsymbol{v},i} \right)^\top \boldsymbol{x} \,.$$

By Lemma 2, we have $\|\boldsymbol{e}_{\boldsymbol{v}}\| < \tau_t \sqrt{(t+1)m}$ with overwhelming probability. Moreover by Lemma 3, we have $\|\boldsymbol{R}_{i,j,\gamma} \cdot \boldsymbol{e}_{\boldsymbol{v},i}\| \leq 12\sqrt{2m} \cdot \|\boldsymbol{e}_{\boldsymbol{v},i}\|$ with overwhelming probability, and since $v_{i,j,\gamma} \in [0, r-1]$ it follows that

$$\left\| \boldsymbol{e}_{\boldsymbol{v},0} + \sum_{i \in [t]} \sum_{j \in [\mu]} \sum_{\gamma=0}^{k} v_{i,j,\gamma} \boldsymbol{R}_{i,j,\gamma} \boldsymbol{e}_{\boldsymbol{v},i} \right\| = O(t \cdot \mu \cdot k \cdot r \cdot \sigma_t \cdot m) \,.$$

Finally, by Lemma 14, the error term has absolute value at most:

$$\left(q\alpha_t \cdot \omega(\sqrt{\log m}) + \sqrt{m}/2\right) \cdot O\left(t \cdot \mu \cdot k \cdot r \cdot \sigma_t \cdot m\right) \,.$$

3.4 Security Reduction

In this section we prove the following theorem.

Theorem 19. *If the decision-*$\mathsf{LWE}_{q,n,\chi}$ *problem is infeasible, then the predicate encryption scheme described above is weak attribute hiding-selective attribute secure.* ▮

Following [2,3], we define additional algorithms. These will not be used in the real scheme, but we need them in our proofs.

$\mathsf{Sim.Setup}(1^n, \boldsymbol{\mu}, \boldsymbol{w}^\star)$. The algorithm choses random $\boldsymbol{A}$, $\boldsymbol{R}^\star_{i,j,k}$ and $\boldsymbol{u}$, it uses $\mathsf{TrapGen}$ to generate $\boldsymbol{B}^\star$ and defines $\boldsymbol{A}_{i,j,\gamma} \leftarrow \boldsymbol{A}\boldsymbol{R}^\star_{i,j\gamma} - r^\gamma w^\star_{i,j} \boldsymbol{B}^\star$. Specifically, on input a security parameter n, an hierarchical format of depth d $\boldsymbol{\mu} = (\ell, d; \mu_1,$

$\dots, \mu_d)$, and a challenge vector $\boldsymbol{w}^\star = (\boldsymbol{w}_1^\star, \dots, \boldsymbol{w}_d^\star)$, the algorithm generates public and secret parameters as follows: Choose random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. For $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \dots, k$, choose uniformly random matrices $\boldsymbol{R}_{i,j,\gamma}^\star \in \mathbb{Z}_q^{n \times m}$. Select a uniformly random vector $\boldsymbol{u} \in \mathbb{Z}_q^n$. Use algorithm $\mathsf{TrapGen}(q, n, m)$ to select a uniformly random $n \times m$-matrix $\boldsymbol{B}^\star \in \mathbb{Z}_q^{n \times m}$ with a basis $\boldsymbol{T}_{\boldsymbol{B}^\star} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^\perp(\boldsymbol{B}^\star)$ such that $\|\widetilde{\boldsymbol{T}_{\boldsymbol{B}^\star}}\| \leq O(\sqrt{n \log q})$. For $i \in [d]$, $j \in [\mu_i]$ and $\gamma = 0, \dots, k$, set $\boldsymbol{A}_{i,j,\gamma} \leftarrow \boldsymbol{A}\boldsymbol{R}_{i,j,\gamma}^\star - r^\gamma w_{i,j}^\star \boldsymbol{B}^\star$.

Output $mpk = (\boldsymbol{A}, \{\boldsymbol{A}_{i,j,\gamma}\}, \boldsymbol{u})$ and $msk = (\{\boldsymbol{R}_{i,j,\gamma}^\star\}, \boldsymbol{B}^\star, \boldsymbol{T}_{\boldsymbol{B}^\star})$.

$\mathsf{Sim.Derive}(mpk, d_{\boldsymbol{v}}, \boldsymbol{v}_t)$. Secret keys are now created by using the trapdoor $\boldsymbol{T}_{\boldsymbol{B}^\star}$, sampled by Sim.Setup, and the SampleBasisRight algorithm. Specifically, on input the master public key mpk, the secret key for the vector $\boldsymbol{v} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_{t-1})$, and the vector $\boldsymbol{v}_t$, the algorithm generates a secret key for the vector $\boldsymbol{v'} = (\boldsymbol{v}_1, \dots, \boldsymbol{v}_t)$ by constructing a short basis for $\Lambda_q^\perp(F_{\boldsymbol{v'}})$, as defined by Equation 2, by invoking $\boldsymbol{S} \leftarrow \mathsf{SampleBasisRight}(\boldsymbol{A}, \boldsymbol{B}_{\boldsymbol{v'}}^\star, \overline{\boldsymbol{R}^\star}, \boldsymbol{T}_{\boldsymbol{B}^\star}, \sigma_t)$. Output $d_{\boldsymbol{v'}} = \boldsymbol{S}$.

$\mathsf{Sim.Enc}(mpk, \boldsymbol{w}, \mathfrak{m})$. The algorithm differs from Enc in the sense that it uses matrices $\boldsymbol{R}_{i,j,\gamma}^\star$ and $\boldsymbol{B}^\star$ instead of matrices $\boldsymbol{R}_{i,j,\gamma}$ and $\boldsymbol{B}$. Specifically, on input master public key mpk, vector $\boldsymbol{w} = (\boldsymbol{w}_1, \dots, \boldsymbol{w}_t)$, and message $\mathfrak{m} \in \{0, 1\}$, the algorithm generates a ciphertext C as follows: Choose a uniformly random vector $\boldsymbol{s} \xleftarrow{\$} \mathbb{Z}_q^n$, a noise vector $\boldsymbol{x} \leftarrow \overline{\Psi}_{\alpha_t}^m$ and a noise term $x \leftarrow \overline{\Psi}_{\alpha_t}$. Compute $\boldsymbol{c}_0 = \boldsymbol{A}^\top \boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^m$. For $i \in [t]$, $j \in [\mu_i]$ and $\gamma = 0, \dots, k$ compute $\boldsymbol{c}_{i,j,\gamma} = (\boldsymbol{A}_{i,j,\gamma} + r^\gamma w_{i,j} \boldsymbol{B}^\star)^\top \boldsymbol{s} + {\boldsymbol{R}^\star}_{i,j,\gamma}^\top \boldsymbol{x} \in \mathbb{Z}_q^m$. Compute $c' = \boldsymbol{u}^\top \boldsymbol{s} + x + \mathfrak{m} \cdot \lfloor q/2 \rceil \in \mathbb{Z}_q$. Output $C = (\boldsymbol{c}_0, \{\boldsymbol{c}_{i,j,\gamma}\}, c')$.

For a probabilistic polynomial-time adversary $\mathcal{A}$, our proof of security will consist of the following sequence of 6 games between $\mathcal{A}$ and $\mathcal{C}$. The six games are defined as follows:

$\mathbf{Game_0}$. $\mathcal{C}$ runs the Setup algorithm, answers $\mathcal{A}$'s secret key queries using the Derive algorithm, and generates the challenge ciphertext using the Enc with vector $\boldsymbol{w}_0$ and message $\mathfrak{m}_0$.

$\mathbf{Game_1}$. In this game $\mathcal{C}$ uses the simulation algorithms. Specifically, $\mathcal{C}$ runs the Sim.Setup algorithm with $\boldsymbol{w}^\star = \boldsymbol{w}_0$, answers $\mathcal{A}$'s secret key queries using the Sim.Derive algorithm, and generates the challenge ciphertext using the Sim.Enc with vector $\boldsymbol{w}_0$ and message $\mathfrak{m}_0$.

$\mathbf{Game_2}$. It is the same as the $\mathbf{Game_1}$ except that the challenge ciphertext is randomly chosen from the ciphertext space.

$\mathbf{Game_3}$. It is the same as the $\mathbf{Game_2}$ except that $\mathcal{C}$ runs the Sim.Setup algorithm with $\boldsymbol{w}^\star = \boldsymbol{w}_1$.

$\mathbf{Game_4}$. It is the same as the $\mathbf{Game_3}$ except that $\mathcal{C}$ generates the challenge ciphertext using the Sim.Enc with vector $\boldsymbol{w}_1$ and message $\mathfrak{m}_1$.

$\mathbf{Game_5}$. $\mathcal{C}$ runs the Setup algorithm, answers $\mathcal{A}$'s secret key queries using the Derive algorithm, and generates the challenge ciphertext using the Enc with vector $\boldsymbol{w}_1$ and message $\mathfrak{m}_1$.

We defer the proof that, for $i = 0, \dots, 4$, $\mathbf{Game_i}$ is indistinguishable form $\mathbf{Game_{i+1}}$ under the appropriate assumptions, to the full version of this paper.

Parameters. From the previous sections we can extract the parameters required for correctness and security of the system.

- We need to ensure that for each $t \in [d]$, correctness holds. Specifically, Lemma 18 requires $q/\log q = \Omega(\sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m^{3/2})$ and $\alpha_t \leq (t \cdot \log q \cdot \sigma_t \cdot \mu \cdot \frac{r}{\log r} \cdot m \cdot \omega(\sqrt{\log m}))^{-1}$.
- By Theorem 6, algorithm TrapGen requires $q > 2$ and $m > 6n \lg q$ to work.
- By Corollary 16, to have algorithm SampleBasisLeft working correctly in the Derive algorithm, we need for each $t \in [d]$, $\sigma_t > \|\widetilde{d_{\boldsymbol{v}}}\| \cdot \omega(\sqrt{\log((t+1)m)})$. Thus, we have $\sigma_t > \sigma_{\mathrm{TG}} \cdot \omega((\log m)^{t/2})$.
- By Corollary 17, to have algorithm SampleBasisRight working correctly in the Sim.Derive algorithm, we need for each $t \in [d]$, $\sigma_t > \|\widetilde{\boldsymbol{T_B}}\| \cdot s_{\overline{\boldsymbol{R}}} \cdot \omega(\sqrt{\log((t+1)m)})$. Thus, by Theorem 6, $\|\widetilde{\boldsymbol{T_B}}\| < \sigma_{\mathrm{TG}}$ and, by Lemma 3, $s_{\overline{\boldsymbol{R}}} = \|\overline{\boldsymbol{R}}\| = O(t \cdot \mu \cdot (\log_r q + 1) \cdot \sqrt{(t+1)m})$ due the particular structure of $\overline{\boldsymbol{R}}$, where $\mu = \max_{i \in [d]} \mu_i$. Thus, $\sigma_t > O(\sqrt{n \log q}) \cdot O(\mu \cdot (\log_r q + 1) \cdot \sqrt{(t+1)m}) \cdot \omega(\sqrt{(t+1)m})$.
- Regev's reduction must apply: $q > 2\sqrt{n}/\alpha_t$

3.5 Wrapping Up

Proof of Theorem 19. From the previous sections and lemmata, we have shown that our hierarchical inner-product encryption is weak attribute hiding-selective attribute secure assuming decision-$\mathsf{LWE}_{q,n,\chi}$ problem is infeasible.

4 Application

4.1 Identity-Based Encryption with Wildcards

One of the main applications of IBE and HIBE schemes is email encryption, where users can encrypt a message to the owner of the email address without having to obtain a certified copy of the owner's public key first. Motivated by the fact that many email addresses correspond to groups of users rather than single individuals, Abdalla *et al.* [1] introduced the concept of identity-based cryptography with wildcards (WIBE). In a WIBE scheme, decryption keys are issued exactly as in a standard HIBE scheme and the main difference lies in the encryption process. More specifically, in a WIBE scheme, the sender can encrypt messages to more general *patterns* consisting of identity strings and *wildcards* so that any identity matching the given pattern can decrypt.

Next we show how to convert any HIPE scheme in a WIBE one by using an encoding first introduced in [13]. This let us to obtain the first WIBE scheme based on lattice assumptions.

Let us start with some notation. A pattern is described by a vector $P = (P_1, \ldots, P_\ell) \in (\{0,1\}^\star \bigcup \{\star\})^\ell$, where $\star$ is a special wildcard symbol. We say that identity $ID = (ID_1, \ldots, ID_{\ell'})$ matches P, denoted $ID \in_\star P$, if and only if $\ell' \leq \ell$ and for all $i = 1, \ldots, \ell'$ we have that $ID_i = P_i$ or $P_i = \star$. Note that under this definition, any ancestor of a matching identity is also a matching identity. This is reasonable for our purposes because any ancestor can derive the secret key of a matching descendant identity anyway. If $P = (P_1, \dots, P_\ell)$ is a pattern, then we define $\mathrm{W}(P)$ to be the set of wildcard positions in P, i.e. $\mathrm{W}(P) = \{1 \le i \le \ell \ : \ P_i = \star\}$. Formally, a WIBE scheme is a tuple of algorithms (Setup, Derive, Enc, Dec) providing the following functionality. The Setup and Derive algorithms behave exactly as those of a HIPE scheme. To create a ciphertext of a message $\mathfrak{m} \in \{0,1\}^\star$ intended for all identities matching pattern P, the sender computes $C \leftarrow \mathsf{Enc}(mpk, P, \mathfrak{m})$. Any of the intended recipients $ID \in_\star P$ can decrypt the ciphertext using its own decryption key as $\mathfrak{m} \leftarrow \mathsf{Dec}(d_{ID}, C)$.

Let HIPE $=$ ($\mathsf{Setup_H}$, $\mathsf{Derive_H}$, $\mathsf{Enc_H}$, $\mathsf{Dec_H}$) be a Hierarchical inner-product encryption. We can construct the scheme WIBE $=$ ($\mathsf{Setup_W}$, $\mathsf{Derive_W}$, $\mathsf{Enc_W}$, $\mathsf{Dec_W}$) as follows:

$\mathsf{Setup_W}(1^\lambda, 1^\ell)$. The algorithm returns the output of $\mathsf{Setup_H}(1^\lambda, \boldsymbol{\mu} = (2\ell, \ell; (\mu_i = 2)_{i \in [\ell]}))$. So the hierarchy $\boldsymbol{\mu}$ is of depth ℓ and each level has dimension 2.

$\mathsf{Derive_W}(msk, ID)$. For a pattern $ID = (ID_1, \dots, ID_\ell)$, the key generation algorithm constructs vector $\boldsymbol{y} \in \Sigma$ by setting, for each $i \in [\ell]$, $\boldsymbol{y}_i = (1, P_i)$. We denote this transformation by $\boldsymbol{y} = \mathsf{KEncode}(ID)$. Then the key generation algorithm returns $d_P = \mathsf{Derive_H}(msk, \boldsymbol{y})$.

$\mathsf{Enc_W}(mpk, P)$. The algorithm constructs vector $\boldsymbol{x} \in \Sigma$ in the following way: For each $i \in [\ell]$ the algorithms sets $\boldsymbol{x}_i = (-r_i \cdot P_i, r_i)$ if $P_i \neq \star$, $\boldsymbol{x}_i = (0, 0)$ otherwise. We denote this transformation by $\boldsymbol{x} = \mathsf{CEncode}(\boldsymbol{z})$. Then the encryption algorithm returns $C = \mathsf{Enc_H}(mpk, \boldsymbol{x})$.

$\mathsf{Dec_W}(d_P, C)$. The algorithm returns the output of $\mathsf{Dec_H}(d_P, C)$.

Correctness. Correctness follows from the observation that for identity ID and pattern P, we have that $f_{\mathsf{KEncode}(ID)}(\mathsf{CEncode}(P)) = 1$ if and only if $ID \in_\star P$.

Security. Let $\mathcal{A}$ be an adversary for WIBE that tries to break the scheme for an hierarchy of depth ℓ and consider the following adversary $\mathcal{B}$ for HIPE that uses $\mathcal{A}$ as a subroutine and tries to break a HIPE with hierarchy format $\boldsymbol{\mu} = (2\ell, \ell; (\mu_i = 2i)_{i \in [\ell]})$ by interacting with challenger $\mathcal{C}$. $\mathcal{B}$ receives a mpk for HIPE and passes it to $\mathcal{A}$. Whenever $\mathcal{A}$ asks for the key for identity ID, $\mathcal{B}$ constructs $\boldsymbol{y} = \mathsf{KEncode}(P)$ and asks $\mathcal{C}$ for a key $d_{\boldsymbol{y}}$ for $\boldsymbol{y}$ and returns it to $\mathcal{A}$. When $\mathcal{A}$ asks for a challenge ciphertext by providing $(\mathfrak{m}_0, P_0^\star)$ and $(\mathfrak{m}_1, P_1^\star)$, $\mathcal{B}$ simply computes $\boldsymbol{x}_0 = \mathsf{CEncode}(P_0^\star)$ and $\boldsymbol{x}_1 = \mathsf{CEncode}(P_1^\star)$ and gives the pair $(\mathfrak{m}_0, \boldsymbol{x}_0), (\mathfrak{m}_1, \boldsymbol{x}_1)$ to $\mathcal{C}$. $\mathcal{B}$ then returns the challenge ciphertext obtained from $\mathcal{C}$ to $\mathcal{A}$. Finally, $\mathcal{B}$ outputs $\mathcal{A}$'s guess. Notice that, $\mathcal{B}$'s simulation is perfect. Indeed, we have that if for all $\mathcal{A}$'s queries satisfy the game constraint, then all $\mathcal{B}$'s queries have the same property. Thus $\mathcal{B}$'s advantage is the same as $\mathcal{A}$'s.

4.2 Chosen-Ciphertext Security

As we have seen in the previous section, given an HIPE scheme is possible to construct a WIBE scheme. Thus, to apply the techniques of [6] to obtain an ℓ-level HIPE scheme secure against chosen-ciphertext attacks we will face the same issues faced by [1].

Thus, following [1], we show that we may use a IND-wAH-sID-HIPE-CPA-secure HIPE of depth $2d + 2$ and a strongly unforgeable signature scheme

$(\mathsf{SigGen}, \mathsf{Sign}, \mathsf{Verify})$ to construct an IND-wAH-sID-HIPE-CCA-secure HIPE of depth d. We adapt the encoding function Encode defined in [1] to the HIPE case in the following way: For a HIPE scheme for hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \ldots, \mu_d)$, we define two encoding functions, one to encode secret keys and one to encode ciphertext. Specifically, for any two values a and b in $\mathbb{Z}_N^\star$ such that $a \neq b$, the encode function for secret keys $\mathsf{KEncode}$ works as follow: $\mathsf{KEncode}(\boldsymbol{v}) = ((1, a), \boldsymbol{v}_1, \ldots, (1, a), \boldsymbol{v}_t)$, for any vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$ with $t \leq d$ and $\mathsf{KEncode}(\boldsymbol{v}, vk) = ((1, a), \boldsymbol{v}_1, \ldots, (1, a), \boldsymbol{v}_t, (1, b), (1, vk))$. On the other hand, the encode function for ciphertext $\mathsf{CEncode}$ works as follow: $\mathsf{CEncode}(\boldsymbol{w}) = ((a, -1), \boldsymbol{w}_1, \ldots, (a, -1), \boldsymbol{w}_t)$, for any vector $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$ with $t \leq d$ and $\mathsf{CEncode}(\boldsymbol{w}, vk) = ((a, -1), \boldsymbol{w}_1, \ldots, (a, -1), \boldsymbol{w}_t, (b, -1), (vk, -1))$.

Construction. Given a HIPE scheme $\mathsf{HIPE} = (\mathsf{Setup}, \mathsf{Derive}, \mathsf{Enc}, \mathsf{Dec})$ for hierarchical format $\boldsymbol{\mu} = (\ell, 2d + 2; 2, \mu_1, 2, \mu_2, \ldots, 2, \mu_d, 2, 2)$, consider the following HIPE scheme $\mathsf{HIPE}' = (\mathsf{Setup}', \mathsf{Derive}', \mathsf{Enc}', \mathsf{Dec}')$ for hierarchical format $\boldsymbol{\mu} = (\ell, d; \mu_1, \mu_2, \ldots, \mu_d)$.

Key Derivation. The secret key for vector $\boldsymbol{v} = (\boldsymbol{v}_1, \ldots, \boldsymbol{v}_t)$, with $t \leq d$ under HIPE' is the secret key corresponding to identity $\mathsf{KEncode}(\boldsymbol{v})$ under HIPE.

Encryption. To encrypt a message $\mathtt{m}$ under a vector $\boldsymbol{w} = (\boldsymbol{w}_1, \ldots, \boldsymbol{w}_t)$ and using a master public key mpk, the following steps are performed: First, generate a signature key pair $(sk, vk) \xleftarrow{\$} \mathsf{SigGen}$. Then compute $C \xleftarrow{\$} \mathsf{Enc}(mpk, \mathsf{CEncode}(\boldsymbol{w}, vk), \mathtt{m})$ and $\sigma \xleftarrow{\$} \mathsf{Sign}(sk, C)$. The final ciphertext is (vk, C, σ).

Decryption. To decrypt a ciphertext (vk, C, σ) using a private key $d_{\boldsymbol{v}}$ for a vector $\boldsymbol{v}$, first check that $\mathsf{Verify}(vk, C, \sigma) = \mathtt{valid}$. If not, output $\perp$. Otherwise, compute $d = \mathsf{Derive}(d_{\boldsymbol{v}}, ((1, b), (1, vk)))$ and output $\mathsf{Dec}(d, C)$. Note that in this case d is the decryption for the identity $\mathsf{KEncode}(\boldsymbol{v}, vk)$ in HIPE.

Proof Sketch. Let $\mathcal{A}$ be an IND-wAH-sID-HIPE-CCA adversary against the HIPE' scheme. Then there exists an IND-wAH-sID-HIPE-CPA attacker $\mathcal{C}$ against HIPE that uses $\mathcal{A}$ as a subroutine. $\mathcal{C}$ can simulate $\mathcal{A}$'s environment in a straight forward way. Then $\mathcal{C}$ wins the game whenever $\mathcal{A}$ does as long as $\mathcal{C}$ does not make any illegal key derivation queries. We will argue this fact briefly. First consider the queries that $\mathcal{C}$ makes to respond to $\mathcal{A}$'s key derivation query $\boldsymbol{v}$. Let $\boldsymbol{v'} = \mathsf{KEncode}(\boldsymbol{v})$ and let $\boldsymbol{w}^{'\star} = \mathsf{CEncode}(\boldsymbol{w}^\star, vk^\star)$. We have the following:

1. If $|\boldsymbol{v'}| > |\boldsymbol{w}^{'\star}|$ then $f_{\boldsymbol{v'}}(\boldsymbol{w}^{'\star}) = 0$.
2. If $|\boldsymbol{v}| = |\boldsymbol{w}^{'\star}|$ then still $f_{\boldsymbol{v'}}(\boldsymbol{w}^{'\star}) = 0$ because $\boldsymbol{v'}$ and $\boldsymbol{w}^{'\star}$ are different on the next to last level ($\boldsymbol{v'}$ contains $(1, a)$ there, while $\boldsymbol{w}^{'\star}$ contains a $(b, -1)$ and they are not orthogonal).
3. If $|\boldsymbol{v'}| < |\boldsymbol{w}^{'\star}|$ then the only way to have $f_{\boldsymbol{v'}}(\boldsymbol{w}^{'\star}) = 1$ is if also $f_{\boldsymbol{v}}(\boldsymbol{w}^\star) = 1$, which are illegal queries in $\mathcal{A}$'s game as well.

Second, consider the key derivation queries that $\mathcal{C}$ makes in order to respond to $\mathcal{A}$'s decryption queries. If $\mathcal{A}$ makes decryption query $(\boldsymbol{v}, (vk, C, \sigma))$, then $\mathcal{C}$ makes a key derivation query for $\boldsymbol{v'} = \mathsf{KEncode}(\boldsymbol{v}, vk)$. Let $\boldsymbol{w}^{'\star} = \mathsf{CEncode}(\boldsymbol{w}^\star, vk^\star)$. Then we have two cases:

1. If $vk \neq vk^\star$ then $f_{\boldsymbol{v}'}(\boldsymbol{w}^{'\star}) = 0$ either because $\boldsymbol{v}$ has a $(1, a)$ where $\boldsymbol{w}^{'\star}$ has a $(b, -1)$, or because they differ on the last level.
2. If $vk = vk^\star$, then we have the following three sub-cases:
   (a) If $|\boldsymbol{v}'| > |\boldsymbol{w}^{'\star}|$ then $f_{\boldsymbol{v}'}(\boldsymbol{w}^{'\star}) = 0$.
   (b) If $|\boldsymbol{v}'| < |\boldsymbol{w}^{'\star}|$ then still $f_{\boldsymbol{v}'}(\boldsymbol{w}^{'\star}) = 0$ because $\boldsymbol{v}'$ and $\boldsymbol{w}^{'\star}$ are different on the next to last level ($\boldsymbol{v}'$ contains $(1, a)$ there, while $\boldsymbol{w}^{'\star}$ contains a $(b, -1)$ and they are not orthogonal).
   (c) If $|\boldsymbol{v}| = |\boldsymbol{w}^{'\star}|$ then the only way to have $f_{\boldsymbol{v}'}(\boldsymbol{w}^{'\star}) = 1$ is if also $f_{\boldsymbol{v}}(\boldsymbol{w}) = 1$ but this case can be proved to have negligible probability under the one-time security of the signature scheme.

Acknowledgments. This work was supported by the French ANR-09-VERS-016 BEST Project, by the Brazilian CAPES Foundation, and by the European Commission through the ICT Program under Contract ICT-2007-216676 ECRYPT II and the FP7-ICT-2011-EU-Brazil Program under Contract 288349 SecFuNet.

References

1. Abdalla, M., Catalano, D., Dent, A.W., Malone-Lee, J., Neven, G., Smart, N.P.: Identity-Based Encryption Gone Wild. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 300–311. Springer, Heidelberg (2006)
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 553–572. Springer, Heidelberg (2010)
3. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional Encryption for Inner Product Predicates from Learning with Errors. In: Lee, D.H. (ed.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 21–40. Springer, Heidelberg (2011)
4. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: ACM STOC Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM Press (May 1996)
5. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. In: STACS 2009, pp. 75–86 (2009)
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. SIAM Journal on Computing 36(5), 1301–1328 (2007)
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public Key Encryption with Keyword Search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
8. Boneh, D., Sahai, A., Waters, B.: Functional Encryption: Definitions and Challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
9. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
10. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 523–552. Springer, Heidelberg (2010)
11. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC Annual ACM Symposium on Theory of Computing, pp. 197–206. ACM Press (May 2008)

12. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., Vimercati, S. (eds.) ACM CCS 2006: 13th Conference on Computer and Communications Security, pp. 89–98. ACM Press (October/November 2006)
13. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
14. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
15. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on Gaussian measures. In: 45th FOCS Annual Symposium on Foundations of Computer Science, pp. 372–381. IEEE Computer Society Press (October 2004)
16. Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner-Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
17. Okamoto, T., Takashima, K.: Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
18. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In: Mitzenmacher, M. (ed.) 41st ACM STOC Annual ACM Symposium on Theory of Computing, pp. 333–342. ACM Press (May/June 2009)
19. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Gabow, H.N., Fagin, R. (eds.) 37th ACM STOC Annual ACM Symposium on Theory of Computing, pp. 84–93. ACM Press (May 2005)
20. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)