2020

Trapdoor Delegation and HIBE from Middle-Product LWE in Standard Model

Quoc Huy Le
University of Wollongong, qhl576@uowmail.edu.au

Dung Hoang Duong
University of Wollongong, hduong@uow.edu.au

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Josef Pieprzyk

Recommended Citation

Le, Quoc Huy; Duong, Dung Hoang; Susilo, Willy; and Pieprzyk, Josef, "Trapdoor Delegation and HIBE from Middle-Product LWE in Standard Model" (2020). *Faculty of Engineering and Information Sciences - Papers: Part B*. 4402.
https://ro.uow.edu.au/eispapers1/4402

Trapdoor Delegation and HIBE from Middle-Product LWE in Standard Model

Abstract

© 2020, Springer Nature Switzerland AG. At CRYPTO 2017, Roşca, Sakzad, Stehlé and Steinfeld introduced the Middle–Product LWE (MPLWE) assumption which is as secure as Polynomial-LWE for a large class of polynomials, making the corresponding cryptographic schemes more flexible in choosing the underlying polynomial ring in design while still keeping the equivalent efficiency. Recently at TCC 2019, Lombardi, Vaikuntanathan and Vuong introduced a variant of MPLWE assumption and constructed the first IBE scheme based on MPLWE. Their core technique is to construct lattice trapdoors compatible with MPLWE in the same paradigm of Gentry, Peikert and Vaikuntanathan at STOC 2008. However, their method cannot directly offer a Hierarchical IBE construction. In this paper, we make a step further by proposing a novel trapdoor delegation mechanism for an extended family of polynomials from which we construct, for the first time, a Hierachical IBE scheme from MPLWE. Our Hierarchy IBE scheme is provably secure in the standard model.

Disciplines

Engineering | Science and Technology Studies

Publication Details

Le, H., Duong, H., Susilo, W. & Pieprzyk, J. (2020). Trapdoor Delegation and HIBE from Middle-Product LWE in Standard Model. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 12146 LNCS 130-149.

Trapdoor Delegation and HIBE from Middle-Product LWE in Standard Model

Huy Quoc Le[1,2(✉)], Dung Hoang Duong[1(✉)], Willy Susilo[1], Josef Pieprzyk[2,3]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong NSW 2522, Australia.
qhl576@uowmail.edu.au, {hduong,wsusilo}@uow.edu.au
[2] CSIRO Data61, Sydney, NSW, Australia.
[3] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.
josef.pieprzyk@data61.csiro.au

Abstract. At CRYPTO 2017, Roşca, Sakzad, Stehlé and Steinfeld introduced the Middle–Product LWE (MPLWE) assumption which is as secure as Polynomial-LWE for a large class of polynomials, making the corresponding cryptographic schemes more flexible in choosing the underlying polynomial ring in design while still keeping the equivalent efficiency. Recently at TCC 2019, Lombardi, Vaikuntanathan and Vuong introduced a variant of MPLWE assumption and constructed the first IBE scheme based on MPLWE. Their core technique is to construct lattice trapdoors compatible with MPLWE in the same paradigm of Gentry, Peikert and Vaikuntanathan at STOC 2008. However, their method cannot directly offer a Hierachical IBE construction. In this paper, we make a step further by proposing a novel trapdoor delegation mechanism for an extended family of polynomials from which we construct, for the first time, a Hierachical IBE scheme from MPLWE. Our Hierachy IBE scheme is provably secure in the standard model.

Key words: Middle–Product LWE, trapdoor, HIBE, standard model, lattices

1 Introduction

Hierarchical identity-based encryption (HIBE) [7, 9] is a variant of IBE [17], which embeds a directed tree. The nodes of the tree are identities and the children identities are produced by appending extra information to their parent identities. HIBEs can be found in many applications such as forward-secure encryption [3], broadcast encryption [5, 19] and access control to pervasive computing information [8] to name a few most popular.

In lattice-based cryptography, a crucial tool for constructing IBE and HIBE schemes is a trapdoor. The GPV construction, for instance, applies *trapdoor preimage sampleable functions* [6]. The trapdoor plays a role of master secret key that is used to sample private key for each identity (following a distribution that is negligibly close to uniform). This trapdoor is applied by Gentry et al. [6]

to construct their IBE from lattices in the random oracle model. Using the same paradigm as [6], Agrawal et al. [1] introduced their IBE scheme in the standard model. Cash et al. [4] define *bonsai tree* with four basic principles in delegating a lattice basis (i.e., delegating a trapdoor in the [6] sense). The bonsai tree technique helps to resolve some open problems in lattice-based cryptography. It allows us to construct some lattice-based primitives in the standard model (without random oracles) as well as it facilitates delegation for purposes such as lattice-based HIBE schemes. At the same time, Agrawal et al. [2] proposed two distinct trapdoor delegations following the definition of trapdoor from [6]. Their techniques have been used to construct a HIBE scheme in the standard model, which is more efficient than the one from [4]. Micciancio and Peikert in their work [13] introduced a simpler and more efficient trapdoor generation and delegation mechanism.

The middle-product learning with errors problem (MPLWE) is a variant of the polynomial learning with error problem (PLWE) proposed by Roşca et al. [16]. It exploits the middle-product of polynomials modulo q. The authors of [16] have proved that MPLWE is as secure as PLWE for a large class of polynomials. This allows more flexibility in choosing underlying polynomial rings when designing cryptosystems. In [16], the authors have constructed a Regev-type public key encryption scheme based on MPLWE, which is as efficient as that built over Ring-LWE [12]. Recently, Lombardi et al. [10] have generalized MPLWE and call it degree-parametrized MPLWE (DMPLWE). They have proved that DMPLWE is as hard as PLWE using similar arguments as in [16]. Further, the authors of [10] have introduced a lattice trapdoor construction (following the trapdoor notion of [13]) for DMPLWE. The construction can be used to design a dual Regev encryption. The dual encryption allows the authors of [10] to come up with IBE constructions in both the random oracle model and the standard model. The standard model IBE in [10] is adapted from the framework of [1]. However, a DMPLWE-based construction for a standard model HIBE cannot be directly obtained from the standard model IBE of [10]. Thus there is a need for more work in order to define an appropriate trapdoor delegation mechanism for the polynomial setting.

Our contribution. In this paper, we follow the line of research initiated by the work [10]. In particular, we introduce a novel technique for delegating lattice trapdoors from DMPLWE and construct a new HIBE scheme based on DMPLWE. Our HIBE scheme is provably secure in the standard model. We follow the framework from [1] and [10].

Let $\overline{\mathbf{a}} = (a_1, \cdots, a_{t'})$ be a t'-family of polynomials. We can interpret any polynomial as a structured matrix, e.g. Toeplitz matrix [15], and hence $\overline{\mathbf{a}}$ can be represented as a concatenated structured matrix, say $\mathbf{A}$. The trapdoor from [10] is a modification of the trapdoor used in [13] and is defined for a family of polynomials. More specifically, in [10], a trapdoor for the family $\overline{\mathbf{a}}$ is a collection td_a of *short* polynomials (here *short* means small coefficients), from which we form a matrix $\mathbf{R}$ such that $\mathbf{A} \cdot \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$, where $\mathbf{G}$ is the concatenated structured matrix of $\overline{\mathbf{g}} = (g_1, \cdots, g_{\gamma\tau})$, namely $g_j = 2^{\eta} x^{d\zeta}$ for $j = \zeta\tau + \eta + 1$ with $\eta \in$

$\{0, \cdots, \tau - 1\}$, $\zeta \in \{0, \cdots, \gamma - 1\}$. We call $\overline{\mathbf{g}}$ the *primitive family.* The trapdoor td_a is used to search for a t'-family of polynomials $\overline{\mathbf{r}} := (r_1, \cdots, r_{t'})$ (following some distribution that is close to uniform) such that $\langle \overline{\mathbf{a}}, \overline{\mathbf{r}} \rangle := \sum_{i=1}^{t'} a_i \cdot r_i = u$ for any given polynomial u of appropriate degree.

For a construction of DMPLWE–based HIBE, we need to derive a trapdoor for an extended family of polynomials, say $\overline{\mathbf{f}} = (\overline{\mathbf{a}}|\overline{\mathbf{h}}) = (a_1, \cdots, a_{t'}|h_1, \cdots, h_{t''})$, from a trapdoor for $\overline{\mathbf{a}}$. To this end, we first proceed with the case $t'' = \gamma\tau$, i.e., the number of polynomials in $\overline{\mathbf{h}}$ has to be the same as the number in $\overline{\mathbf{g}}$. We transform $\overline{\mathbf{h}}$ into a matrix $\mathbf{H}$, and then apply the idea of trapdoor delegation from [13] to obtain the trapdoor td_f for $\overline{\mathbf{f}}$. We generalize the trapdoor delegation to the case $t'' = m\gamma\tau$, a multiple of $\gamma\tau$ for $m \geq 1$.

Using the proposed polynomial trapdoor delegation, we build the first HIBE based on DMPLWE, which is provably INDr–sID–CPA secure in the standard model. To produce a private key for an identity $\mathsf{id} = (id_1, \cdots, id_\ell)$ at depth ℓ, we form an *extended* family $\overline{\mathbf{f}}_{\mathsf{id}} = (\overline{\mathbf{a}}, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$ in which each $\overline{\mathbf{h}}^{(i,\mathsf{bit})} = (h_1^{(i,\mathsf{bit})}, \cdots, h_{t'}^{(i,\mathsf{bit})})$ is a family of random polynomials. Then our trapdoor delegation helps to get a trapdoor for $\overline{\mathbf{f}}_{\mathsf{id}}$, which plays the role of the private key with respect to the identity id. Deriving a private key for a child identity $\mathsf{id}|id_{\ell+1} = (id_1, \cdots, id_\ell, id_{\ell+1})$ from a parent identity $\mathsf{id} = (id_1, \cdots, id_\ell)$ is done in similar way by appending $\overline{\mathbf{h}}^{(\ell+1,id_{\ell+1})}$ to $\overline{\mathbf{f}}_{\mathsf{id}}$ so we get $\overline{\mathbf{f}}_{\mathsf{id}|id_{\ell+1}} = (\overline{\mathbf{a}}, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)}, \overline{\mathbf{h}}^{(\ell+1,id_{\ell+1})})$. Then we use the trapdoor delegation to get its private key from the private key (trapdoor) of $\overline{\mathbf{f}}_{\mathsf{id}}$. In order for the security proof to work, we need to put a condition on t' such that t' is a multiple of $\gamma\tau$. Indeed, the condition ensures that the simulator is able to simulate an answer to a private key query of the adversary. The answer is generated using a trapdoor for some $\overline{\mathbf{h}}^{(i,id_i)}$, both of which are not chosen randomly but produced by a trapdoor generator.

Open problems. Our trapdoor delegation technique is restricted to the relation of the number of polynomials in the primitive family $\overline{\mathbf{g}}$ and the number of polynomials in the extended family (i.e., $t'' = m\gamma\tau$, a multiple of $\gamma\tau$). It would be interesting and might be useful to find a new trapdoor delegation method that could be applied for an arbitrary $t'' \geq 1$. Moreover, if we had another mechanism that would help to find a trapdoor for $\overline{\mathbf{f}}'_{\mathsf{id}}$, where $\overline{\mathbf{f}}'_{\mathsf{id}} = (\overline{\mathbf{a}}, \langle \overline{\mathbf{h}}^{(1,id_1)}, \overline{\mathbf{b}} \rangle, \cdots, \langle \overline{\mathbf{h}}^{(\ell,id_\ell)}, \overline{\mathbf{b}} \rangle)$, given a random $\overline{\mathbf{b}}$ and its trapdoor td_b, then we might be able to apply the HIBE framework of [2] to get a smaller ciphertext size than that of our work here. One more question is that whether or not there exists a trapdoor (and delegation) method that does not utilise the Toeplitz representation but applies directly polynomials, with a relevant definition of polynomial trapdoor.

Organisation. In Section 2, we review some related background. The trapdoor delegation mechanism for polynomials in MPLWE setting will be presented in

Section 3. We will give an MPLWE-based HIBE construction in the standard model in Section 4. Section 5 concludes this work.

2 Preliminaries

Notations. We denote by $R^{<n}[x]$ the set of polynomials of degree less than n with coefficients in a commutative ring R. We mainly work with the rings of polynomials over $\mathbb{Z}$ such as $\mathbb{Z}[x]$ and $\mathbb{Z}_q[x]$. We use italic small letters for polynomials in R. For a positive integer ℓ, $[\ell]$ stands for the set $\{1, 2, \cdots, \ell\}$. The Gram-Schmidt orthogonal matrix of a matrix $\mathbf{A}$ is written as $\tilde{\mathbf{A}}$. We call $\overline{\mathbf{h}}$ an n-family (or n-vector) of polynomials if $\overline{\mathbf{h}} = (h_1, \cdots, h_n)$, where h_i's are polynomials. By $\overline{\mathbf{a}}|\overline{\mathbf{h}}$, we denote a concatenated (or expanded) family, which consists of all ordered polynomials from both $\overline{\mathbf{a}}$ and $\overline{\mathbf{h}}$. For two n-families of polynomials $\overline{\mathbf{a}} = (a_1, \cdots, a_n)$ and $\overline{\mathbf{r}} = (r_1, \cdots, c_n)$, their scalar product is defined as $\langle \overline{\mathbf{a}}, \overline{\mathbf{r}} \rangle := \sum_{i=1}^{n} a_i \cdot r_i$. The notation $\mathcal{U}(X)$ stands for the uniform distribution over the set X. The Euclidean and sup norms of a vector $\mathbf{u}$ (as well as a matrix) are written as $\|\mathbf{u}\|$ and $\|\mathbf{u}\|_\infty$, respectively.

2.1 IBE and HIBE: Syntax and Security

Syntax. An IBE system [17] is a tuple of algorithms $\{\mathsf{Setup}, \mathsf{Extract}, \mathsf{Encrypt}, \mathsf{Decrypt}\}$, in which: (1) $\mathsf{Setup}(1^n)$ on input a security parameter 1^n, outputs a master public key MPK and a master secret key MSK; (2) $\mathsf{Extract}(\mathsf{MSK}, \mathsf{id})$ on input the master secret key MSK and an identity id, outputs a private key $\mathsf{SK}_{\mathsf{id}}$; (3) $\mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}, \mu)$ on input the master public key MPK, an identity id and a message μ, outputs a ciphertext CT; and (4) $\mathsf{Decrypt}(\mathsf{id}, \mathsf{SK}_{\mathsf{id}}, \mathsf{CT})$ on input an identity id and its associated private key $\mathsf{SK}_{\mathsf{id}}$ and a ciphertet CT, outputs a message μ.

A HIBE [7] is a tuple of algorithms $\{\mathsf{Setup}, \mathsf{Extract}, \mathsf{Derive}, \mathsf{Encrypt}, \mathsf{Decrypt}\}$, where Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$ are defined in similar way as for IBE. Let λ be the maximum depth of identities. An identity at depth $\ell \leq \lambda$ is represented by a binary vector $\mathsf{id} = (id_1, \cdots, id_\ell) \in \{0, 1\}^\ell$ of dimension ℓ and it is considered as the "parent" of the appended $\mathsf{id}|id_{\ell+1} = (id_1, \cdots, id_\ell, id_{\ell+1})$. The algorithm $\mathsf{Setup}(1^n, 1^\lambda)$ needs a slight modification as it accepts both n and λ as the input. For the input: private key $\mathsf{SK}_{\mathsf{id}}$ and $\mathsf{id}|\mathsf{id}_{\ell+1}$, the algorithm $\mathsf{Derive}(\mathsf{SK}_{\mathsf{id}}, \mathsf{id}|\mathsf{id}_{\ell+1})$ outputs the private key $\mathsf{SK}_{\mathsf{id}|id_{\ell+1}}$ for the identity $\mathsf{id}|id_{\ell+1}$. If we consider the master secret key as the private key for any identity at depth 0, then Derive has the same function as $\mathsf{Extract}$. (H)IBE has to be correct in the following sense:

$$\Pr[\mathsf{Decrypt}(\mathsf{id}, \mathsf{SK}_{\mathsf{id}}, \mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}, \mu))] = 1 - \mathsf{negl}(n),$$

where the probability is taken over random coin tosses for Setup, $\mathsf{Extract}$, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$ (for IBE) and Derive (for HIBE).

Security. For the purpose of our paper, we present the following security game for INDr-sID-CPA or indistinguishability of ciphertexts under a selective chosen-identity and adaptive chosen-plaintext attack. In the game, the adversary has to announce his target identity at the very beginning. For a security parameter n, let $\mathcal{M}_n$ and $\mathcal{C}_n$ be the plaintext and ciphertext spaces, respectively. The game consists of six phases as follows:

- **Initialize:** The challenger chooses a maximum depth λ and gives it to the adversary. The adversary outputs a target identity $\mathsf{id}^* = (id_1^*, \cdots, id_k^*), (k \leq \lambda)$.
- **Setup:** The challenger runs $\mathsf{Setup}(1^n, 1^\lambda)$ and sends the public parameters MPK to the adversary. The master secret key MSK is kept secret by the challenger.
- **Queries 1:** The adversary makes private key queries adaptively. The queries are for identities id of the form $\mathsf{id} = (id_1, \cdots, id_m)$ for some $m \leq \lambda$, which are not a prefix of id^*. This is to say that $\mathsf{id}_i \neq \mathsf{id}_i^*$ for all $i \in [m]$ and $m \leq k$. The challenger answers the private key query for id by calling the private key extraction algorithm $\mathsf{Extract}$ and sends the key to the adversary.
- **Challenge:**
  - Whenever the adversary decides to finish Queries 1, he will output the challenge plaintext $\mu^* \in \mathcal{M}_n$.
  - The challenger chooses a random bit $b \in \{0, 1\}$. It computes the challenge ciphertext CT^*. If $b = 0$, it calls the encryption algorithm and gets $\mathsf{CT}^* \leftarrow \mathsf{Encrypt}(\mathsf{MPK}, \mathsf{id}^*, \mu^*)$. If $b = 1$, it chooses a random $CT \in \mathcal{C}_n$ so $\mathsf{CT}^* \leftarrow CT$. CT^* is then sent to the adversary.
- **Queries 2:** The adversary makes the private key queries again and the challenger answers the queries as in **Queries 1**.
- **Guess:** The adversary outputs a guess $b' \in \{0, 1\}$ and he wins if $b' = b$.

The adversary in the above game is referred to as an INDr-sID-CPA adversary. The advantage of an adversary $\mathcal{A}$ in the game is $\mathsf{Adv}^{\mathsf{HIBE},\lambda,\mathcal{A}}(n) = |\Pr[b = b'] - 1/2|$.

Definition 1 (INDr-sID-CPA). *A depth λ HIBE system $\mathcal{E}$ is selective-identity indistinguishable from random if for any probabilistic polynomial time (PPT) INDr-sID-CPA adversary $\mathcal{A}$, the function $\mathsf{Adv}^{\mathsf{HIBE},\lambda,\mathcal{A}}(n)$ is negligible. We say that $\mathcal{E}$ is secure for the depth λ.*

2.2 Lattices and Gaussian Distributions

For positive integers n, m, q and a matrix $\mathbf{A} \in \mathbb{Z}^{n \times m}$, We consider lattices $\Lambda_q^\perp(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = \mathbf{0} \pmod q\}$ $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{z} \in \mathbb{Z}^m : \mathbf{Az} = \mathbf{u} \pmod q)$. If $\Lambda_q^{\mathbf{u}}(\mathbf{A}) \neq \emptyset$ then $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ is a shift of $\Lambda_q^\perp(\mathbf{A})$. Specifically, if there exists $\mathbf{e}$ such that $\mathbf{Ae} = \mathbf{u} \pmod q$ then $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \Lambda_q^\perp(\mathbf{A}) + \mathbf{e}$.

Definition 2 (Gaussian Distribution). *Given countable set $S \subset \mathbb{R}^n$ and $\sigma > 0$, the discrete Gaussian distribution $D_{S,\sigma,\mathbf{c}}$ over S centered at some $\mathbf{c} \in$*

S with standard deviation σ is defined as $\mathcal{D}_{S,\sigma,\mathbf{c}}(\mathbf{x}) := \rho_{\sigma,\mathbf{c}}(\mathbf{x})/\rho_{\sigma,\mathbf{c}}(S)$, where $\rho_{\sigma,\mathbf{c}}(\mathbf{x}) := \exp(\frac{-\pi\|\mathbf{x}-\mathbf{v}\|^2}{\sigma^2})$ and $\rho_{\sigma,\mathbf{c}}(S) := \sum_{\mathbf{x}\in S}\rho_{\sigma,\mathbf{c}}(\mathbf{x})$. If $\mathbf{c} = \mathbf{0}$, we simply write ρ_σ and $D_{S,\sigma}$ instead of $\rho_{\sigma,\mathbf{0}}$, $D_{S,\mathbf{0},\sigma}$, respectively.

We use of the following tail bound of $D_{\Lambda,\sigma}$ for parameter σ sufficiently larger than the *smothing parameter* $\eta_\epsilon(\Lambda)$, defined to be the the smallest real number s such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \epsilon$; cf. [14].

Lemma 1 ([6, Lemma 2.9]). *For any $\epsilon > 0$, any $\sigma \geq \eta_\epsilon(\mathbb{Z})$, and any $K > 0$, we have $\Pr_{x\leftarrow D_{\mathbb{Z},\sigma,c}}[|x-c| \geq K\cdot\sigma] \leq 2e^{-\pi K^2}\cdot\frac{1+\epsilon}{1-\epsilon}$. In particular, if $\epsilon \in (0,\frac{1}{2})$ and $K \geq \omega(\sqrt{\log n})$, then the probability that $|x-c| \geq K\cdot\sigma$ is negligible in n.*

2.3 Degree-Parametrized Middle-Product Learning with Errors

Definition 3 (Middle-Product, [16, Definition 3.1]). *Let d_a, d_b, k, d be integers such that $d_a + d_b - 1 = 2k + d$. We define the middle-product of two polynomials $a \in \mathbb{Z}^{<d_a}[x]$ and $b \in \mathbb{Z}^{<d_b}[x]$ as follows:*

$$\odot_d : \mathbb{Z}^{<d_a}[x] \times \mathbb{Z}^{<d_b}[x] \to \mathbb{Z}^{<d}[x], (a,b) \mapsto \left\lfloor \frac{ab \mod x^{k+d}}{x^k} \right\rfloor. \tag{1}$$

Lemma 2 ([16, Lemma 3.3]). *Let $d, k, n > 0$. For all $r \in R^{<k+1}[x]$, $a \in R^{<n}[x]$, $s \in R^{<n+d+k-1}[x]$, it holds that $r \odot_d (a \odot_{d+k} s) = (r\cdot a) \odot_d s$.*

Definition 4 (DMPLWE, [10, Definition 9]). *Let $n' > 0$, $q \geq 2$, $\mathbf{d} = (d_1, \cdots, d_{t'}) \in [\frac{n'}{2}]^{t'}$, and let χ be a distribution over $\mathbb{R}_q$. For $s \in \mathbb{Z}_q^{<n'-1}[x]$, we define the distribution $\mathsf{DMP}_{q,n',\mathbf{d},\chi}(s)$ over $\prod_{i=1}^{t'}(\mathbb{Z}_q^{n'-d_i}[x] \times \mathbb{R}_q^{d_i}[x])$ as follows:*

- *For each $i \in [t']$, sample $f_i \xleftarrow{\$} \mathbb{Z}_q^{<n'-d_i}[x]$ and sample $e_i \leftarrow \chi^{d_i}[x]$ (represented as a polynomial of degree less than d_i).*
- *Output $(f_i, \mathsf{ct}_i := f_i \odot_{d_i} s + e_i)_{i\in[t']}$.*

The degree-parametrized MPLWE (named $\mathsf{DMPLWE}_{q,n',\mathbf{d},\chi}$) requires to distinguish between arbitrarily many samples from $\mathsf{DMP}_{q,n',\mathbf{d},\chi}(s)$ and the same number of samples from $\prod_{i=1}^{t'}\mathcal{U}(\mathbb{Z}_q^{n'-d_i}[x] \times \mathbb{R}_q^{d_i}[x])$.

For $\mathcal{S} > 0$, let $\mathcal{F}(\mathcal{S}, \mathbf{d}, n')$ be the set of monic polynomials f in $\mathbb{Z}[x]$ with the constant coeficient coprime with q, that have degree $m \in \cap_{i=1}^{t'}[d_i, n'-d_i]$ and satisfy $\mathsf{EF}(f) < \mathcal{S}$. For a polynomial $f \in \mathbb{Z}[x]$ of degree m, $\mathsf{EF}(f)$ is the *expansion factor* ([11]) of f defined as follows: $\mathsf{EF}(f) := \max_{g\in\mathbb{Z}^{<2m-1}[x]}\frac{\|g \mod f\|_\infty}{\|g\|_\infty}$. Following [16], Lombardi et al. [10] showed that DMPLWE is as hard as $\mathsf{PLWE}_{q,\chi}^{(f)}$ (defined below) for any polynomial f of $\mathsf{poly}(n')$-bounded expansion factor.

Definition 5 (PLWE, [18]). *Let $n > 0$, $q \geq 2$, f be a polynomial of degree m, χ be a distribution over $\mathbb{R}[x]/f$. The decision problem $\mathsf{PLWE}_{q,\chi}^{(f)}(s)$ is to distinguish between arbitrarily many samples $\{(a, a\cdot s + e) : a \xleftarrow{\$} \mathbb{Z}_q[x]/f, e \leftarrow \chi\}$, and the same number of samples from $\mathcal{U}(\mathbb{Z}_q[x]/f \times \mathbb{R}_q[x]/f)$ over the randomness of $s \xleftarrow{\$} \mathbb{Z}_q[x]/f$.*

It is proven that $\mathsf{PLWE}^{(f)}_{q,\chi}(s)$ is as hard as solving Shortest Vector Problem (SVP) over ideal lattices in $\mathbb{Z}[x]/f$; see [18] for more detail.

Theorem 1 (Hardness of DMPLWE, [10, Theorem 2]). *Let $n' > 0$, $q \geq 2$, $\mathbf{d} = (d_1, \cdots, d_{t'}) \in [\frac{n'}{2}]^{t'}$, and $\alpha \in (0,1)$. Then, there exists a probabilistic polynomial time (PPT) reduction from $\mathsf{PLWE}^{(f)}_{q,D_{\alpha\cdot q}}$ for any polynomial f in $\mathcal{F}(\mathcal{S}, \mathbf{d}, n')$ to $\mathsf{DMPLWE}_{q,n',\boldsymbol{d},D_{\alpha'\cdot q}}$ with $\alpha' = \alpha\mathcal{S}\sqrt{\frac{n'}{2}}$.*

2.4 Lattice Trapdoor Generation for DMPLWE

Definition 6 (G-Trapdoor, [13, Definition 5.2]). *Let $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{G} \in \mathbb{Z}_q^{n\times m'}$ be matrices with $m \geq m' \geq n$. A matrix $\mathbf{R} \in \mathbb{Z}^{(m-m')\times m'}$ is called* **G***-trapdoor for* $\mathbf{A}$ *with tag* $\mathbf{H}$ *(which is an invertible matrix in $\mathbb{Z}_q^{n\times n}$) if $\mathbf{A} \cdot \begin{bmatrix}\mathbf{R}\\ \mathbf{I}_{m'}\end{bmatrix} =$* $\mathbf{HG}$.

In particular, it is suggested in [13, Section 4] that $\mathbf{G} = \mathbf{I}_n \otimes \begin{bmatrix}1 \ 2 \cdots 2^k\end{bmatrix}$. We can choose $\mathbf{H} = \mathbf{I}_n$ or such that $\mathbf{HG}$ is any (column) permutation of $\mathbf{G}$ which is similar to the usage of $\mathbf{G}$ in [10]. In fact, it is defined in [10, Definition 11]) that $\mathbf{A} \in \mathbb{Z}^{k\times(m+k\tau)}$ and $\mathbf{G} := \mathbf{I}_k \otimes [1 \ 2 \cdots 2^{\tau-1}] \in \mathbb{Z}_q^{k\times k\tau}$. However, $\mathbf{G}$ is used in $\mathsf{SamplePre}$ (see below) is actually a (column) permutation of $\mathbf{I}_k \otimes [1 \ 2 \cdots 2^{\tau-1}]$ from which the authors can extracts polynomial g_i in $\overline{\mathbf{g}}$ thanks to the Toeplitz representation of polynomials (see Equation (6)). We first recall their definition and some basic properties.

Definition 7 (Toeplitz matrix). *Let R be a ring and $d, k > 0$ be integers. For any polynomial $u \in R^{<n}[x]$, we define the Topelitz matrix $\mathsf{Tp}^{n,d}(u)$ for u as a matrix in $R^{(n+d-1)\times d}$ whose the i-th column is the coefficient vector of $x^{i-1}\cdot u$ arranged in increasing degree of x with 0 inserted if any.*

By Definition 7, it is easy to assert the following Lemma.

Lemma 3. *Let $u \in \mathbb{Z}^{<n}[x]$. Then,*

$$\mathsf{Tp}^{n,d}(u) = [\mathsf{Tp}^{n+d-1,1}(u)|\mathsf{Tp}^{n+d-1,1}(x\cdot u)|\cdots|\mathsf{Tp}^{n+d-1,1}(x^{d-1}\cdot u)].$$

Lemma 4 ([10, Lemma 7]). *For positive integers k, n, d and polynomials $u \in R^{<k}[x]$, if $v \in R^{<n}[x]$, then $\mathsf{Tp}^{k,n+d-1}(u)\cdot\mathsf{Tp}^{n,d}(v) = \mathsf{Tp}^{k+n-1,d}(u\cdot v)$.*

Theorem 2 ([10, Theorem 4]). *Let $\mathbf{G} := \mathbf{I}_k \otimes \begin{bmatrix}1 \ 2 \cdots 2^{\tau-1}\end{bmatrix} \in \mathbb{Z}_q^{k\times k\tau}$ and matrices $\mathbf{A} \in \mathbb{Z}^{k\times(m+k\tau)}$, $\mathbf{R} \in \mathbb{Z}^{m\times k\tau}$ be such that $\mathbf{A}\cdot\left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}_{k\tau}\end{smallmatrix}\right] = \mathbf{G}$. Then, there exists an efficient algorithm $\mathcal{P} = (\mathcal{P}_1, \mathcal{P}_2)$ that executes according to the two following phases:*

- ***offline:*** *$\mathcal{P}_1(\mathbf{A}, \mathbf{R}, \sigma)$ performs some polynomial-time preprocessing on input $(\mathbf{A}, \mathbf{R}, \sigma)$ and outputs a state* st.

– ***online****: for a given vector* $\mathbf{u}$*,* $\mathcal{P}_2(\mathsf{st}, \mathbf{u})$ *samples a vector from* $D_{\Lambda^\perp_u(\boldsymbol{A}),\sigma}$ *as long as*

$$\sigma \geq \omega(\sqrt{\log k}) \cdot \sqrt{7(s_1(\mathbf{R})^2 + 1)}, \tag{2}$$

where $s_1(\mathbf{R}) := \max_{\|\mathbf{u}\|=1} \|\mathbf{Ru}\|$ *is the largest singular value of* $\mathbf{R}$*.*

The value $s_1(\mathbf{R})$ is upper bounded as explained by the lemma given below.

Lemma 5 ([10, Lemma 6]). *For any matrix* $\mathbf{R} = (R_{ij}) \in \mathbb{R}^{m\times n}$*,*

$$s_1(\mathbf{R}) \leq \sqrt{mn} \cdot \max_{i,j} |R_{ij}|. \tag{3}$$

G-Trapdoor for a family of polynomials. We recap the construction of lattice trapdoors for DMPLWE from [10]. The construction applies two PPT algorithms $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$. Suppose that $q = \mathsf{poly}(n)$, $d \leq n$, $dt/n = \Omega(\log n)$, $d\gamma = n + 2d - 2$, $\tau := \lceil \log_2 q \rceil$, $\beta := \lceil \frac{\log_2(n)}{2} \rceil \ll q/2$, and σ satisfies Equation (7) below. Then, $\mathsf{TrapGen}$ and $\mathsf{SamplePre}$ work as follows:

$\underline{\mathsf{TrapGen}(1^n)}$: On input a security parameter n, do the following:

- Sample $\overline{\mathbf{a}}' = (a_1, \cdots, a_t) \xleftarrow{\$} (\mathbb{Z}_q^{<n}[x])^t$, and for all $j \in [\gamma\tau]$, sample $\overline{\mathbf{w}}^{(j)} = (w_1^{(j)}, \cdots, w_t^{(j)}) \leftarrow (\Gamma^d[x])^t$ where $\Gamma = \mathcal{U}(\{-\beta, \cdots, \beta\})$.
- For all $j \in [\gamma\tau]$, define $u_j = \langle \overline{\mathbf{a}}', \overline{\mathbf{w}}^{(j)} \rangle$ and $a_{t+j} = g_j - u_j$, where

$$g_j = 2^\eta x^{d\zeta} \in \mathbb{Z}_q^{n+d-1}[x], \tag{4}$$

for $j = \zeta\tau + \eta + 1$ with $\eta \in \{0, \cdots, \tau - 1\}$, $\zeta \in \{0, \cdots, \gamma - 1\}$. Set $\overline{\mathbf{g}} := (g_1, \cdots, g_{\gamma\tau})$.
- Output $\overline{\mathbf{a}} := (a_1, \cdots, a_t, a_{t+1}, \cdots, a_{t+\gamma\tau})$ with its corresponding trapdoor $\mathsf{td} := (\overline{\mathbf{w}}^{(1)}, \cdots, \overline{\mathbf{w}}^{(\gamma\tau)})$.

The amount of space to store the trapdoor td is $O(d(\gamma\tau)t) = O(n\tau t)$ as $d\gamma = n + 2d - 2 \leq 3n$.

$\underline{\mathsf{SamplePre}(\overline{\mathbf{a}} = (a_1, \cdots, a_{t+\gamma\tau}), \mathsf{td} = (\overline{\mathbf{w}}^{(1)}, \cdots, \overline{\mathbf{w}}^{(\gamma\tau)}), u, \sigma)}$: On input a family $\overline{\mathbf{a}}$ of $t + \gamma\tau$ polynomials together with its trapdoor td_ϵ generated by $\mathsf{TrapGen}$, and a polynomial u of degree less than $n + 2d - 2$, do the following:

- First, construct (implicitly) matrices $\mathbf{A}', \mathbf{A}, \mathbf{T}, \mathbf{G}$ for $\overline{\mathbf{a}}'$, $\overline{\mathbf{a}}$, td, $\overline{\mathbf{g}}$, respectively:

$$\mathbf{A}' = [\mathsf{Tp}^{n,2d-1}(a_1)|\cdots|\mathsf{Tp}^{n,2d-1}(a_t)],$$

$$\mathbf{A} = [\mathsf{Tp}^{n,2d-1}(a_1)|\cdots|\mathsf{Tp}^{n,2d-1}(a_t)|\mathsf{Tp}^{n+d-1,d}(a_{t+1})|\cdots|\mathsf{Tp}^{n+d-1,d}(a_{t+\gamma\tau})],$$

$$\mathbf{T} = \begin{bmatrix} \mathsf{Tp}^{d,d}(w_1^{(1)}) \cdots \mathsf{Tp}^{d,d}(w_1^{(\gamma\tau)}) \\ \vdots \qquad\qquad \vdots \\ \mathsf{Tp}^{d,d}(w_t^{(1)}) \cdots \mathsf{Tp}^{d,d}(w_t^{(\gamma\tau)}) \end{bmatrix} \in \mathbb{Z}_q^{(2d-1)t\times d\gamma\tau}, \tag{5}$$

$$\mathbf{G} = [\mathsf{Tp}^{n+d-1,d}(g_1)|\cdots|\mathsf{Tp}^{n+d-1,d}(g_{\gamma\tau})] \in \mathbb{Z}_q^{d\gamma\times d\gamma\tau}, \tag{6}$$

$$\mathbf{I}_{d\gamma\tau} = \begin{bmatrix} \mathsf{Tp}^{1,d}(1) \cdots & \\ \cdots & \cdots \\ & \cdots \mathsf{Tp}^{1,d}(1) \end{bmatrix} \in \mathbb{Z}_q^{d\gamma\tau \times d\gamma\tau}.$$

Then $\mathbf{A} = [\mathbf{A}'|\mathbf{G} - \mathbf{A}'\mathbf{T}]$ and hence $\mathbf{A} \cdot \left[\begin{smallmatrix} \mathbf{T} \\ \mathbf{I}_{d\gamma\tau} \end{smallmatrix} \right] = \mathbf{G}$. Recall that $d\gamma = n + 2d - 2$.

- The polynomial u is represented it as $\mathbf{u} = \mathsf{Tp}^{n+2d-2,1}(u) \in \mathbb{Z}_q^{n+2d-2}$.
- Sample vector $\mathbf{r} \in \mathbb{Z}^{(2d-1)t+d\gamma\tau}$ from $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}),\sigma}$ using the trapdoor $\mathbf{T}$ in means of [13], where

$$\sigma \geq \omega(\sqrt{\log(d\gamma)}) \cdot \sqrt{7(s_1(\mathbf{T})^2 + 1)}, \tag{7}$$

and

$$s_1(\mathbf{T}) \leq \sqrt{(2d-1)t \cdot (d\gamma\tau)} \cdot \beta. \tag{8}$$

- Split $\mathbf{r}$ into $\mathbf{r} = [\mathbf{r}_1^\top | \cdots | \mathbf{r}_{t+\gamma\tau}^\top]^\top$, and rewrite it (in column) as a Toeplitz matrix of polynomials $r_1, \cdots, r_{t+\gamma\tau}$, where $\mathbf{r}_j = \mathsf{Tp}^{2d-1,1}(r_j)$, $\deg(r_j) < 2d-1$, $\forall j \in [t]$, $\mathbf{r}_j = \mathsf{Tp}^{d,1}(r_j)$, $\deg(r_{t+j}) < d, \forall j \in t+1, \cdots, t+\gamma\tau$.
- Output $\overline{\mathbf{r}} := (r_1, \cdots, r_{t+\gamma\tau})$. Note that, $\langle \overline{\mathbf{a}}, \overline{\mathbf{r}} \rangle = \sum_{i=1}^{t+\gamma\tau} a_i \cdot r_i = u$; see [10, Section 5] for more details.

The runtime of $\mathsf{SamplePre}$ is $\tilde{O}(nt)$ and the output distribution of (r_i) is exactly the conditional distribution

$$(D_{\mathbb{Z}^{2d-1},\sigma}[x])^t \times (D_{\mathbb{Z}^d,\sigma}[x])^{\gamma\tau} | \sum_{i=1}^{t+\gamma\tau} a_i \cdot r_i = u.$$

Further on, we give our main results, which are a trapdoor delegation mechanism useful for extending a family of polynomials as well as a HIBE system built using the framework of [1]. From now on, by "trapdoor", we mean "**G**-trapdoor", where **G** is defined by Equation (6). Also, we denote the output of $\mathsf{TrapGen}$ by $\overline{\mathbf{a}}_\epsilon$ and td_ϵ and call them *the root family* and *the root trapdoor*, respectively. The Toeplitz matrices $\mathbf{A}_\epsilon$ and $\mathbf{T}_\epsilon$ correspond to $\overline{\mathbf{a}}_\epsilon$ and td_ϵ, respectively.

3 Trapdoor Delegation for Polynomials

3.1 Description

In order to exploit the trapdoor technique in constructing a MPLWE-based HIBE scheme, we have to solve the problem of delegating a trapdoor (in the sense of Definition 6) for $\overline{\mathbf{f}} = (a_1, \cdots, a_{t'} | h_1, \cdots, h_{t''})$ provided the trapdoor for $\overline{\mathbf{a}} = (a_1, \cdots, a_{t'})$. As mentioned in Section 2.4, we can represent $\overline{\mathbf{f}}$ as a concatenation of Toeplitz matrices of the form $\mathbf{F} = [\mathbf{A}|\mathbf{H}]$ in which $\mathbf{A}, \mathbf{H}$ are the Toeplitz representations for $\overline{\mathbf{a}}$ and $\overline{\mathbf{h}} := (h_1, \cdots, h_{t''})$, respectively.

Following Definition 6, our task is to find a matrix $\mathbf{R}$, which satisfies the equation $\mathbf{F} \cdot \left[\begin{smallmatrix} \mathbf{R} \\ \mathbf{I} \end{smallmatrix} \right] = \mathbf{G}$, where $\mathbf{G}$ as given by Equation (6). Recall that, in matrix setting in [13, Section 5.5], this task can be easily done by finding $\mathbf{R}$ that satisfies the relation $\mathbf{AR} = \mathbf{G} - \mathbf{H}$, when we know a trapdoor for $\mathbf{A}$ and $\mathbf{H}$ has the same dimension as $\mathbf{G}$. In our setting, this task is not straightforward. The main reason for this is that the matrices $\mathbf{A}$, $\mathbf{G}$, $\mathbf{H}$ are Toeplitz ones. To be able to apply the idea of trapdoor delegation of [13] to our setting, we have to design $\mathbf{H}$ such that $\mathbf{U} := \mathbf{G} - \mathbf{H}$ is still in the Toeplitz form of some polynomials. In other words, the form of $\mathbf{H}$ should be similar in form and in dimension to that of $\mathbf{G}$ in (6), namely,

$$\mathbf{H} = [\mathsf{Tp}^{n+d-1,d}(h_1)|\cdots|\mathsf{Tp}^{n+d-1,d}(h_{\gamma\tau})] \in \mathbb{Z}_q^{d\gamma\times d\gamma\tau}. \tag{9}$$

This requires that $t'' = \gamma\tau$ and $\deg(h_i) < n + d - 1$ for all $i \in [\gamma\tau]$. If this is the case, the last step is to try to follow [10] using $\mathsf{SamplePre}$ to have $\mathbf{R}$ satisfy $\mathbf{AR} = \mathbf{U}$ given $\mathbf{A}$ and a trapdoor for $\mathbf{A}$. Note that in our polynomial setting $\mathbf{R}$ should be a structured matrix, which can be easily converted into appropriate polynomials r_i.

By generalization, we come up with the following theorem in which $t' = t + k\gamma\tau$ and $t'' = m\gamma\tau$ for $k \geq 1, m \geq 1$:

Theorem 3 (Trapdoor Delegation). *Let n be a positive integer, $q = \mathit{poly}(n)$ be a prime, and d, t, γ, τ, k, m be positive integers such that $d \leq n$, $dt/n = \Omega(\log n)$, $d\gamma = n + 2d - 2$, $k \geq 1$, $m \geq 1$. Let $\tau := \lceil \log_2 q \rceil$ and $\beta := \lceil \frac{\log_2 n}{2} \rceil$. Let $\mathbf{G}$ be matrix as in (6) and $\overline{\mathbf{a}} = (a_1, \cdots, a_{t+k\gamma\tau})$ be a $(t + k\gamma\tau)$-family of polynomials and its associated trapdoor td_a, where $a_i \in \mathbb{Z}_q^{<n}[x]$ for $i \in [t]$ and $a_i \in \mathbb{Z}_q^{<n+d-1}[x]$ for $t + 1 \leq i \leq t + k\gamma\tau$. Suppose that $\overline{\mathbf{h}} = (h_1, \cdots, h_{m\gamma\tau})$ is a $m\gamma\tau$-family of polynomials in $\mathbb{Z}_q^{<n+d-1}[x]$ and $\overline{\sigma} = (\sigma_{k+1}, \cdots, \sigma_{k+m})$ to be determined. Then, there exists an efficient (PPT) algorithm, $\mathsf{SampleTrap}(\overline{\mathbf{a}}, \overline{\mathbf{h}}, \mathsf{td}_a, \overline{\sigma})$ that outputs a trapdoor td_f for $\overline{\mathbf{f}} = (a_1, \cdots, a_{t+k\gamma\tau}|h_1, \cdots, h_{m\gamma\tau})$. Moreover, the amount of space to store the trapdoor td_f is $O(((2d-1)t+(k+m-1)\gamma\tau)\cdot d\gamma\tau) = O(n^2 \log^2 n) = \widetilde{O}(n^2)$.*

3.2 Elementary Trapdoor Delegation

In this section, we present in detail the basic trapdoor delegation for the family $\overline{\mathbf{f}} = (a_1, \cdots, a_{t+\gamma\tau}|h_1, \cdots, h_{\gamma\tau})$ given the root trapdoor td_ϵ for the root family $\overline{\mathbf{a}}_\epsilon = (a_1, \cdots, a_{t+\gamma\tau})$. They are generated by $\mathsf{TrapGen}$, i.e., $\mathsf{SampleTrap}$ for $k = 1$ and $m = 1$. This process is called $\mathsf{TrapDel}$ and is shown as Algorithm 1.

Note that $\mathsf{TrapGen}$, $\overline{\mathbf{a}}_\epsilon = (a_1, \cdots, a_t, a_{t+1}, \cdots, a_{t+\gamma\tau}) \in (\mathbb{Z}_q^{<n}[x])^t \times (\mathbb{Z}_q^{<n+d-1}[x])^{\gamma\tau}$, and the corresponding concatenated Toeplitz matrix $\mathbf{A}_\epsilon \in \mathbb{Z}_q^{(n+2d-2)\times[(2d-1)t+d\gamma\tau]}$ is constructed as

$$\mathbf{A}_\epsilon = [\mathsf{Tp}^{n,2d-1}(a_1)|\cdots|\mathsf{Tp}^{n,2d-1}(a_t)|\mathsf{Tp}^{n+d-1,d}(a_{t+1})|\cdots|\mathsf{Tp}^{n+d-1,d}(a_{t+\gamma\tau})]. \tag{10}$$

The matrix $\mathbf{G}$ has the following form:

$$\mathbf{G} = [\mathsf{Tp}^{n+d-1,d}(g_1)|\cdots|\mathsf{Tp}^{n+d-1,d}(g_{\gamma\tau})],$$

where $g_j = 2^\eta x^{d\zeta}$ for $j = \zeta\tau + \eta + 1$ with $\eta \in \{0, \cdots, \tau - 1\}$, $\zeta \in \{0, \cdots, \gamma - 1\}$. As discussed above, we construct $\mathbf{H} = [\mathsf{Tp}^{n+d-1,d}(h_1)| \cdots |\mathsf{Tp}^{n+d-1,d}(h_{\gamma\tau})]$ for $h_1, \cdots, h_{\gamma\tau}$, whose $\deg(h_i) < n + d - 1$ for all $i \in [\gamma\tau]$. Then the Toeplitz matrix for $\overline{\mathbf{f}}$ takes the form

$$\begin{aligned}\mathbf{F} &= [\mathbf{A}_\epsilon|\mathbf{H}] \\ &= [\mathsf{Tp}^{n,2d-1}(a_1)| \cdots |\mathsf{Tp}^{n,2d-1}(a_t)|\mathsf{Tp}^{n+d-1,d}(a_{t+1})| \cdots |\mathsf{Tp}^{n+d-1,d}(h_{\gamma\tau})].\end{aligned} \tag{11}$$

and

$$\mathbf{G} - \mathbf{H} = [\mathsf{Tp}^{n+d-1,d}(g_1 - h_1)| \cdots |\mathsf{Tp}^{n+d-1,d}(g_{\gamma\tau} - h_{\gamma\tau})]. \tag{12}$$

For $i = 1, \cdots, \gamma\tau$, let $u_i = g_i - h_i$. From Lemma 3, we have

$$\begin{aligned}\mathbf{G} - \mathbf{H} =& [\mathsf{Tp}^{n+2d-2,1}(u_1)| \cdots |\mathsf{Tp}^{n+2d-2,1}(x^{d-1} \cdot (u_1))| \\ & \cdots |\mathsf{Tp}^{n+2d-2,1}(u_{\gamma\tau})| \cdots |\mathsf{Tp}^{n+2d-2,1}(x^{d-1} \cdot (u_{\gamma\tau}))] \\ =& [\mathsf{Tp}^{n+2d-2,1}(v_1)| \cdots |\mathsf{Tp}^{n+2d-2,1}(x^{d-1} \cdot (v_{d\gamma\tau}))],\end{aligned}$$

where $v_i = x^\alpha u_\beta$ for $i = \alpha + d(\beta - 1) + 1$, with $\alpha \in \{0, \cdots, d - 1\}$, $\beta \in \{1, \cdots, \gamma\tau\}$. Let $\mathbf{v}^{(i)} := \mathsf{Tp}^{n+2d-2,1}(v_i)$. Now, for $i = 1, \cdots, \gamma\tau$ we have to find $\mathbf{R} = [\mathbf{r}^{(1)}| \cdots |\mathbf{r}^{(d\gamma\tau)}]$ such that $\mathbf{A}_\epsilon[\mathbf{r}^{(1)}| \cdots |\mathbf{r}^{(d\gamma\tau)}] = [\mathbf{v}^{(1)}| \cdots |\mathbf{v}^{(d\gamma\tau)}]$, which is equivalent to $\mathbf{A}_\epsilon \mathbf{r}^{(i)} = \mathbf{v}^{(i)}$ for $1 \leq i \leq d\gamma\tau$. This can be done using $\mathsf{SamplePre}(\overline{\mathbf{a}}_\epsilon, \mathsf{td}_\epsilon, v_i, \sigma)$. Eventually, we get $\mathbf{r}^{(i)} \in \mathbb{Z}^{(2d-1)t+d\gamma\tau}$, which is sampled from $\mathcal{D}_{\Lambda^\perp_{\mathbf{v}^{(i)}}(\mathbf{A}),\sigma}$, where $\sigma \geq \omega(\sqrt{\log(d\gamma)}) \cdot \sqrt{7((2d - 1)t \cdot (d\gamma\tau) \cdot \beta^2 + 1)}$; see (7), (8).

Finally, we obtain the trapdoor $\mathsf{td}_f = (\overline{\mathbf{r}}^{(1)}, \cdots, \overline{\mathbf{r}}^{(d\gamma\tau)})$ for $\overline{\mathbf{f}}$, where $\overline{\mathbf{r}}^{(i)} = (r_1^{(i)}, \cdots, r_{t+\gamma\tau}^{(i)})$, with $\deg(r_j^{(i)}) < 2d - 1$ for $j \in [t]$, $\deg(r_{t+j}^{(i)}) < d$ for $j \in [\gamma\tau]$ and for all $i \in [d\gamma\tau]$. and its corresponding matrix representation is

$$\mathbf{R} = (R_{ij}) = \begin{bmatrix} \mathsf{Tp}^{2d-1,1}(r_1^{(1)}) & \cdots & \mathsf{Tp}^{2d-1,1}(r_1^{(d\gamma\tau)}) \\ \vdots & & \vdots \\ \mathsf{Tp}^{2d-1,1}(r_t^{(1)}) & \cdots & \mathsf{Tp}^{2d-1,1}(r_t^{(d\gamma\tau)}) \\ \mathsf{Tp}^{d,1}(r_{t+1}^{(1)}) & \cdots & \mathsf{Tp}^{d,1}(r_{t+1}^{(d\gamma\tau)}) \\ \vdots & & \vdots \\ \mathsf{Tp}^{d,1}(r_{t+\gamma\tau}^{(1)}) & \cdots & \mathsf{Tp}^{d,1}(r_{t+\gamma\tau}^{(d\gamma\tau)}) \end{bmatrix} \in \mathbb{Z}^{((2d-1)t+d\gamma\tau)\times d\gamma\tau}. \tag{13}$$

Certainly, we have $\mathbf{F} \cdot \left[\begin{smallmatrix}\mathbf{R}\\ \mathbf{I}\end{smallmatrix}\right] = \mathbf{G}$. Remark that, by Lemma 1,

$$|R_{ij}| \leq \omega(\log n) \cdot \sigma \text{ with probability } 1 - \mathsf{negl}(n). \tag{14}$$

Hence, from Lemma 5

$$s_1(\mathbf{R}) \leq \sqrt{((2d-1)t + d\gamma\tau) \cdot (d\gamma\tau)} \cdot \omega(\log n) \cdot \sigma, \tag{15}$$

where σ satisfies Equation (7).

```
Algorithm 1 $\mathsf{TrapDel}(\overline{\mathbf{a}}, \overline{\mathbf{h}}, \mathsf{td}, \sigma)$
Input: A $(t + k\gamma\tau)$-family of polynomials $\overline{\mathbf{a}} = (a_1, \cdots, a_t, a_{t+1}, \cdots, a_{t+k\gamma\tau}) \in (\mathbb{Z}_q^{<n}[x])^t \times (\mathbb{Z}_q^{<n+d-1}[x])^{k\gamma\tau}$, and its trapdoor $\mathsf{td}_a$, and a $\gamma\tau$-family of polynomials $\overline{\mathbf{h}} = (h_1, \cdots, h_{\gamma\tau}) \in (\mathbb{Z}_q^{<n+d-1}[x])^{\gamma\tau}$, and (implicitly) $\overline{\mathbf{g}} = (g_1, \cdots, g_{\gamma\tau}) \in (\mathbb{Z}_q^{<n+d-1}[x])^{\gamma\tau}$ as in (4).
Output: The trapdoor $\mathsf{td}_f$ for $\overline{\mathbf{f}} = (a_1, \cdots, a_{t+k\gamma\tau}, h_1, \cdots, h_{\gamma\tau})$.
Compute $\overline{\mathbf{u}} = (u_1, \cdots, u_{\gamma\tau}) \leftarrow \overline{\mathbf{g}} - \overline{\mathbf{h}} = (g_1 - h_1, \cdots, g_{\gamma\tau} - h_{\gamma\tau})$.
Define $v_i = x^{\alpha} u_{\beta}$ for $i = \alpha + d(\beta - 1) + 1$, with $\alpha \in \{0, \cdots, d-1\}$, $\beta \in \{1, \cdots, \gamma\tau\}$.
For $i \in [d\gamma\tau]$, call $\mathsf{GenSamplePre}(\overline{\mathbf{a}}, \mathsf{td}_a, v_i, \sigma)$ to get $\overline{\mathbf{r}}^{(i)} = (r_1^{(i)}, \cdots, r_{t+k\gamma\tau}^{(i)})$, where $\deg(r_j^{(i)}) < 2d - 1$ for $j \in [t]$, $\deg(r_{t+j}^{(i)}) < d$ for $j \in [k\gamma\tau]$ and for all $i \in [d\gamma\tau]$.
Return $\mathsf{td}_f = (\overline{\mathbf{r}}^{(1)}, \cdots, \overline{\mathbf{r}}^{(d\gamma\tau)})$.
```

Note that, after having the trapdoor for $\overline{\mathbf{f}}$ and by assigning $\overline{\mathbf{a}}_{\epsilon} \leftarrow \overline{\mathbf{f}}$, $\mathbf{A}_{\epsilon} \leftarrow \mathbf{F}$, we can perform the same procedure explained above. So we get a trapdoor for $\overline{\mathbf{f}}' = (a_1, \cdots, a_{t+\gamma\tau}, h_1, \cdots, h_{\gamma\tau} | z_1, \cdots, z_{\gamma\tau})$ for some $\overline{\mathbf{z}} = (z_1, \cdots, z_{\gamma\tau})$, where $z_i \in \mathbb{Z}_q^{<n+d-1}[x]$. Consequently, we come up with a PPT algorithm called TrapDel (Algorithm 1) in which we consider the expanded families of the form $\overline{\mathbf{a}} = (a_1, \cdots, a_t, a_{t+1}, \cdots, a_{t+k\gamma\tau}) \in (\mathbb{Z}_q^{<n}[x])^t \times (\mathbb{Z}_q^{<n+d-1}[x])^{k\gamma\tau}$ for $k \geq 1$. Also note that TrapDel does not call SamplePre. Instead, it calls a slightly modified variant presented below.

Generalized SamplePre. Accordingly to the expansion of trapdoors, we slightly modify SamplePre in Section 2.4 and call it GenSamplePre. The algorithm works not only with TrapGen (i.e., $k = 1$) but also with TrapDel (i.e., $k > 1$). GenSamplePre is the same as SamplePre except for $k > 1$, where matrices $\mathbf{R}$ given as the input trapdoors are of form (13), while for $k = 1$, the matrix $\mathbf{R}$ is of form (5). If we execute GenSamplePre for input $(\overline{\mathbf{a}} = (a_1, \cdots, a_{t+k\gamma\tau}), \mathsf{td}_a = (\overline{\mathbf{r}}^{(1)}, \cdots, \overline{\mathbf{r}}^{(d\gamma\tau)}), u, \sigma)$, where $\overline{\mathbf{r}}^{(i)} = (r_1^{(i)}, \cdots, r_{k\gamma\tau}^{(i)})$ (with $k > 1$), then td_a should be interpreted as a $((2d-1)t + (k-1)d\gamma\tau) \times d\gamma\tau$-matrix, say $\mathbf{R}^{(k-1)}$, of the form (13). The last row is indexed by $t + (k-1)\gamma\tau$.

3.3 SampleTrap

SampleTrap mentioned in Theorem 3 is described as follows:

$\mathsf{SampleTrap}(\overline{\mathbf{a}} = (a_1, \cdots, a_{t+k\gamma\tau}), \overline{\mathbf{h}} = (h_1, \cdots, h_{m\gamma\tau}), \mathsf{td}_a, \overline{\sigma} = (\sigma_{k+1}, \cdots, \sigma_{k+m}))$:

- **Input:** A $(t+k\gamma\tau)$-family of polynomials $\overline{\mathbf{a}} = (a_1, \cdots, a_t, a_{t+1}, \cdots, a_{t+k\gamma\tau}) \in (\mathbb{Z}_q^{<n}[x])^t \times (\mathbb{Z}_q^{<n+d-1}[x])^{k\gamma\tau}$, its trapdoor td_a and a $m\gamma\tau$-family of polynomials $\overline{\mathbf{h}} = (h_0, \cdots, h_{m\gamma\tau}) \in (\mathbb{Z}_q^{<n+d-1}[x])^{m\gamma\tau}$, where $m \geq 1$, and (implicitly) $\overline{\mathbf{g}} = (g_1, \cdots, g_{\gamma\tau}) \in (\mathbb{Z}_q^{<n+d-1}[x])^{\gamma\tau}$ as in (4).
- **Output:** The trapdoor td_f for $\overline{\mathbf{f}} = (a_1, \cdots, a_{t+k\gamma\tau} | h_1, \cdots, h_{m\gamma\tau})$.
- **Execution:**
  1. Split $\overline{\mathbf{h}} = (\overline{\mathbf{h}}^{(1)}, \cdots, \overline{\mathbf{h}}^{(m)})$ where each $\overline{\mathbf{h}}^{(i)}$ is a $\gamma\tau$-family of polynomials.

```
2. $\mathsf{td}^{(1)} \leftarrow \mathsf{td}_a$, $\overline{\mathbf{a}}^{(1)} \leftarrow \overline{\mathbf{a}}$.
3. For $i = 1$ up to $m$ do:
   • $\mathsf{td}^{(i+1)} \leftarrow \mathsf{TrapDel}(\overline{\mathbf{a}}^{(i)}, \overline{\mathbf{h}}^{(i)}, \mathsf{td}^{(i)}, \sigma_i)$.
   • $\overline{\mathbf{a}}^{(i+1)} \leftarrow (\overline{\mathbf{a}}^{(i)}, \overline{\mathbf{h}}^{(i)})$.
4. Return $\mathsf{td}_f = \mathsf{td}^{(m+1)}$.
```

Let us make few observations for SampleTrap.

Trapdoor $\mathbf{td}_f$. From Section 3.2, we can easily generalize to see that the output td_f is $(\overline{\mathbf{r}}^{(1)}, \cdots, \overline{\mathbf{r}}^{(d\gamma\tau)})$ in which for $i \in [d\gamma\tau]$, $\overline{\mathbf{r}}^{(i)} = (r_1^{(i)}, \cdots, r_{t+(k+m-1)\gamma\tau}^{(i)})$ and $r_j^{(i)} \in \mathbb{Z}_q^{<n+d-1}[x]$ for $j \in [t]$, and $r_{t+j}^{(i)} \in \mathbb{Z}_q^{<d}[x]$ for $j \in [(k+m-1)\gamma\tau]$. We can imply that the matrix representation, named $\mathbf{R}^{(k+m-1)}$, for the trapdoor td_f has the form (13), with the last row's index $t + (k+m-1)\gamma\tau$.

Setting Gaussian parameters $\overline{\boldsymbol{\sigma}} = (\boldsymbol{\sigma_1}, \cdots, \boldsymbol{\sigma_m})$. Note that the algorithm $\mathsf{SamplePre}(\overline{\mathbf{a}}, \mathsf{td}_a, u, \overline{\sigma})$ has to satisfy Condition (7) for each σ_i. The same condition must hold for GenSamplePre. From Equation (13), we can see that the trapdoor $\mathsf{td}^{(i+1)}$ in SampleTrap can be interpreted as a matrix $\mathbf{R}^{(i+1)}$ of dimension $((2d-1)t + (k+i-1)d\gamma\tau) \times (d\gamma\tau)$. Thus, σ_i in Equations (14) and (15) should satisfy $\sigma_i \geq \omega(\sqrt{\log(d\gamma)}) \cdot \sqrt{7(s_1(\mathbf{R}^{(i-1)})^2 + 1)}$, and

$$s_1(\mathbf{R}^{(i-1)}) \leq \sqrt{((2d-1)t + (k+i-1)d\gamma\tau) \cdot (d\gamma\tau)} \cdot \omega(\log n) \cdot \sigma_{i-1}, \text{ where } i \in [m].$$

4 DMPLWE-based HIBE in Standard Model

In this section, we describe a HIBE system based on the DMPLWE problem. Our HIBE scheme is IND-sID-CPA secure in the standard model and is inspired by the construction of IBE from [1]. Note that the authors of [10] use a similar approach. However, the private key $\mathsf{SK}_{\mathsf{id}}$ (with respect to an identity id) in the standard model IBE of [10] is actually not a trapdoor. Therefore, it seems difficult to construct HIBE using this approach. In our HIBE construction, the private key for an identity $\mathsf{id} = (id_1, \cdots, id_\ell)$ of depth ℓ is a trapdoor for a family of polynomials, which corresponds to the public key. So we can derive the private key for the appended identity $\mathsf{id}|id_k = (id_1, \cdots, id_\ell, \cdots id_k)$ using the trapdoor delegation presented in Section 3, where $k > \ell$.

4.1 Construction

Our construction, named HIBE, consists of a tuple of algorithms {Setup, Extract, Derive, Encrypt, Decrypt}. They are described below.

Setup$(1^\lambda, 1^n)$: On input the security parameter n, the maximum depth λ, perform the following:

- Set common parameters as follows:
  - $q = q(n)$ be a prime; d, k be positive integers such that $2d + k \leq n$ and $\frac{n+2d-2}{d}$ is also a positive integer, say γ, i.e., $d\gamma = n + 2d - 2$; $\beta := \lceil \frac{\log_2 n}{2} \rceil$, $\tau := \lceil \log_2 q \rceil$, t is a positive integer and let $t' = t + \gamma\tau$, and plaintext space $\mathcal{M} := \{0,1\}^{<k+2}[x]$.. Note that we will set $t' = m\gamma\tau$ (with $m \geq 2$), that is t is a multiple of $\gamma\tau$ so as to we can apply the trapdoor delegation.
  - For Gaussian parameters used in $\mathsf{Encrypt}$: choose $\overline{\alpha} = (\alpha_1, \cdots, \alpha_\lambda) \in \mathbb{R}^{\lambda}_{>0}$; for Gaussian parameters used in $\mathsf{Extract}$ and Derive: choose $\overline{\Sigma} = (\overline{\sigma}^{(1)}, \cdots, \overline{\sigma}^{(\lambda)})$, where $\overline{\sigma}^{(\ell)} = (\sigma^{(\ell)}_1, \cdots, \sigma^{(\ell)}_m) \in \mathbb{R}^{m}_{>0}$. For $\ell \in [\lambda]$, let $\overline{\Sigma}^{(\ell)} = (\overline{\sigma}^{(1)}, \cdots, \overline{\sigma}^{(\ell)})$; for Gaussian parameters used in $\mathsf{Decrypt}$: choose $\overline{\Psi} = (\Psi_1, \cdots, \Psi_\lambda) \in \mathbb{R}^{\lambda}_{>0}$.

  They all are set as in Section 4.2.
- For $\ell \in [\lambda]$, let $\chi_\ell := \lfloor D_{\alpha_\ell \cdot q} \rceil$ be the rounded Gaussian distribution.
- Use $\mathsf{TrapGen}(1^n)$ to get a root family $\overline{\mathbf{a}}_\epsilon = (a_1, \cdots, a_{t'})$ and its associated root trapdoor td_ϵ.
- Select uniformly a random polynomial $u_0 \in \mathbb{Z}_q^{<n+2d-2}[x]$.
- For each $i \in [\lambda]$, and each $\mathsf{bit} \in \{0,1\}$, sample randomly $\overline{\mathbf{h}}^{(i,\mathsf{bit})} = (h_1^{(i,\mathsf{bit})}, \cdots, h_{t'}^{(i,\mathsf{bit})})$, where each $h_j^{(i,\mathsf{bit})} \in \mathbb{Z}_q^{<n}[x]$ for $j \in [t]$, and each $h_j^{(i,\mathsf{bit})} \in \mathbb{Z}_q^{<n+d-1}[x]$ for $j \in \{t+1, \cdots, t+\gamma\tau\}$. Let $\mathsf{HList} = \{(i, \mathsf{bit}, \overline{\mathbf{h}}^{(i,\mathsf{bit})}) : i \in [\lambda], \mathsf{bit} \in \{0,1\}\}$ be the ordered set of all $\overline{\mathbf{h}}^{(i,\mathsf{bit})}$.
- Set the master secret key $\mathsf{MSK} := \mathsf{td}_\epsilon$.

We denote $\mathsf{id} = (id_1, \cdots, id_\ell) \in \{0,1\}^\ell$ as an identity of depth $\ell \leq \lambda$. All following algorithms will always work on $\overline{\mathbf{a}}_\epsilon = (a_1, \cdots, a_{t'})$ and HList.

Derive$(\mathsf{id}|id_{\ell+1}, \mathsf{SK}_{\mathsf{id}})$: On input $\mathsf{id} = (id_1, \cdots, id_\ell)$, $\mathsf{id}|id_{\ell+1} = (id_1, \cdots, id_\ell, id_{\ell+1})$, private key $\mathsf{SK}_{\mathsf{id}} := \mathsf{td}_{\mathsf{id}}$– the trapdoor for $\overline{\mathbf{f}}_{\mathsf{id}} = (\overline{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$, execute:

1. Output $\mathsf{SK}_{\mathsf{id}|id_{\ell+1}} \leftarrow \mathsf{SampleTrap}(\overline{\mathbf{f}}_{\mathsf{id}}, \overline{\mathbf{h}}^{(\ell+1,id_{\ell+1})}, \mathsf{SK}_{\mathsf{id}}, \overline{\Sigma}^{(\ell+1)})$.

Extract$(\mathsf{id}, \mathsf{MSK})$: On input $\mathsf{id} = (id_1, \cdots, id_\ell)$, $\mathsf{MSK} = \mathsf{td}_\epsilon$, execute:

1. Build $\overline{\mathbf{h}}_{\mathsf{id}} = (\overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$.
2. Output $\mathsf{SK}_{\mathsf{id}} \leftarrow \mathsf{SampleTrap}(\overline{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}_{\mathsf{id}}, \mathsf{MSK}, \overline{\Sigma}^{(\ell)})$.

Encrypt$(\mathsf{id}, \mu, u_0, \alpha_\ell)$: On input $\mathsf{id} = (id_1, \cdots, id_\ell)$, $\mu \in \mathcal{M}$, u_0, α_ℓ, execute:

1. Build $(f_1, \cdots, f_{t'(\ell+1)}) \leftarrow \overline{\mathbf{f}}_{\mathsf{id}} = (\overline{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}^{(1,id_1)} \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$.
2. Sample $s \xleftarrow{\$} \mathbb{Z}_q^{<n+2d+k-1}[x]$.
3. Sample $e_0 \leftarrow \chi_\ell^{k+1}[x]$, compute: $\mathsf{CT}_0 = u_0 \odot_{k+2} s + 2e_0 + \mu$.
4. For $i = 0$ to ℓ do:
   - For $j \in [t]$, sample $e_{i \cdot t' + j} \leftarrow \chi_\ell^{2d+k}[x]$, and compute:
     $$\mathsf{ct}_i = f_{i \cdot t' + j} \odot_{2d+k} s + 2e_{i \cdot t' + j}.$$
   - For $t + 1 \leq j \leq t + \gamma\tau$, sample $e_{i \cdot t' + j} \leftarrow \chi_\ell^{d+k+1}[x]$, and compute:
     $$\mathsf{ct}_i = f_{i \cdot t' + j} \odot_{d+k+1} s + 2e_{i \cdot t' + j}.$$

5. Set $\mathsf{CT}_1 = (\mathsf{ct}_1, \cdots, \mathsf{ct}_{t'(\ell+1)})$, and output ciphertext $\overline{\mathsf{CT}} = (\mathsf{CT}_0, \mathsf{CT}_1)$.

$\underline{\mathbf{Decrypt}(\mathsf{id}, \mathsf{SK}_{\mathsf{id}}, \overline{\mathsf{CT}}, u_0, \Psi_\ell)}$: On input $\mathsf{id} = (id_1, \cdots, id_\ell)$, $\mathsf{SK}_{\mathsf{id}} := \mathsf{td}_{\mathsf{id}}$–the trapdoor for $\bar{\mathbf{f}}_{\mathsf{id}} = (\bar{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$, ciphertext $\overline{\mathsf{CT}} = (\mathsf{CT}_0, \mathsf{CT}_1)$, u_0, and Ψ_ℓ, do:

1. Parse $(f_1, \cdots, f_{t'(\ell+1)}) \leftarrow \bar{\mathbf{f}}_{\mathsf{id}} = (\bar{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$.
2. Sample $\bar{\mathbf{r}} = (r_1, \cdots, r_{t'(\ell+1)}) \leftarrow \mathsf{GenSamplePre}(\bar{\mathbf{f}}_{\mathsf{id}}, \mathsf{SK}_{\mathsf{id}}, u_0, \Psi_\ell)$, i.e., $\langle \bar{\mathbf{f}}_{\mathsf{id}}, \bar{\mathbf{r}} \rangle = \sum_1^{t'(\ell+1)} r_i \cdot f_i = u_0$.
3. Parse $(\mathsf{CT}_0, \mathsf{CT}_1 = (\mathsf{ct}_1, \cdots, \mathsf{ct}_{t'(\ell+1)})) \leftarrow \overline{\mathsf{CT}}$.
4. Output $\mu = (\mathsf{CT}_0 - \sum_{i=1}^{t'(\ell+1)} \mathsf{ct}_i \odot_{k+2} r_i \mod q) \mod 2$.

4.2 Correctness and Parameters

Lemma 6 (Correctness). *For $\ell \in [\lambda]$, if*

$$\alpha_\ell < \frac{1}{4}\left[t'(\ell+1) \cdot (k+1) \cdot \omega(\log n) \cdot \Psi_\ell + \omega(\sqrt{\log n})\right]^{-1}, \qquad (16)$$

then the scheme is correct with probability $1 - \mathsf{negl}(n)$.

Proof. For $\mathsf{id} = (id_1, \cdots, id_\ell)$, we need to show that

$$\mathsf{Decrypt}(\mathsf{id}, \mathsf{SK}_{\mathsf{id}}, \mathsf{Encrypt}(\mathsf{id}, \mu, u_0, \alpha_\ell), u_0, \Psi_\ell) = \mu,$$

with probability $1 - \mathsf{negl}(n)$ over the randomness of Setup, Derive, Extract, Encrypt. Suppose that $\overline{\mathsf{CT}} := (\mathsf{CT}_0, \mathsf{CT}_1 = (\mathsf{ct}_1, \cdots, \mathsf{ct}_{t'(\ell+1)})) \leftarrow \mathsf{Encrypt}(\mathsf{id}, \mu, u_0, \alpha_\ell)$. By Lemma 2, we have

$\mathsf{CT}_0 - \sum_{i=1}^{t'(\ell+1)} \mathsf{ct}_i \odot_{k+2} r_i = \mu + 2(e_0 - \sum_1^{t'(\ell+1)} r_i \odot_{k+2} e_i)$.

Hence, if $\|\mu + 2(e_0 - \sum_1^{t'(\ell+1)} r_i \odot_{k+2} e_i)\|_\infty < q/2$ then μ is recovered.

Therefore, we need to bound the coefficients of $e_0 - \sum_1^{t'(\ell+1)} r_i \odot_{k+2} e_i$. First, note that,

- for $i \in [t]$: $\deg(r_i) < d_r := 2d - 1$, $\deg(e_i) < d_e := k + 1$.
- for $i \in \{t+1, \cdots, t'(\ell+1)\}$: $\deg(r_i) < d_r := d$, $\deg(e_i) < d_e := d + k + 1$.

In gerneral, $d_e + d_r - 1 = 2(d-1) + (k+2)$. Let $\mathbf{r}_i = (\mathbf{r}_{i,0}, \cdots, \mathbf{r}_{i,d_r-1}), \mathbf{e}_i = (\mathbf{e}_{i,0}, \cdots, \mathbf{e}_{i,d_e-1})$ be the vectors of coefficients of r_i and e_i, respectively. By definition of the middle product, $r_i \odot_{k+2} e_i = \sum_{j+w=d-1}^{d+k} \mathbf{r}_{i,j} \cdot \mathbf{e}_{i,w} \cdot x^{j+w}$. By Lemma 1, $\Pr[\|\mathbf{r}_i\|_\infty > \omega(\sqrt{\log n}) \cdot \Psi_\ell] = \mathsf{negl}(n)$, $\Pr[\|\mathbf{e}_i\|_\infty > \omega(\sqrt{\log n}) \cdot \alpha_\ell \cdot q] = \mathsf{negl}(n)$.

Hence $\|r_i \odot_{k+2} e_i\|_\infty < (k+2) \cdot \omega(\log n) \cdot \Psi_\ell \cdot \alpha_\ell \cdot q$. As a result,

$$\left\| e_0 - \sum_1^{t'(\ell+1)} r_i \odot_{k+2} e_i \right\|_\infty \le [t'(\ell+1) \cdot (k+2) \cdot \omega(\log n) \cdot \Psi_\ell + \omega(\sqrt{\log n})] \cdot \alpha_\ell \cdot q.$$

In order for the decryption to be correct, we need Condition (16). □

Setting Parameters. We set the parameters as described below:

- Security prameter n, $q = \mathsf{poly}(n)$ prime, $\beta := \lceil \frac{\log_2(n)}{2} \rceil \ll q/2$, $\tau := \lceil \log_2(q) \rceil$, $\tau = \Theta(\log q) = \Theta(\log n)$, $t' = t + \gamma\tau = m\gamma\tau$ (for some $m \geq 2$), $d \leq n$, $dt/n = \Omega(\log n)$, and $d\gamma = n + 2d - 2 \leq 3n$.
- We set Gaussian parameters used in $\mathsf{Extract}$ and Derive as follows: Recall that, for $\ell \in [\lambda]$, $\overline{\Sigma}^{(\ell)} = (\overline{\sigma}^{(1)}, \cdots, \overline{\sigma}^{(\ell)})$, where each $\overline{\sigma}^{(i)} = (\sigma_1^{(i)}, \cdots, \sigma_m^{(i)}) \in \mathbb{R}^m_{>0}$. It suffices to consider the maximal case happening in $\mathsf{Extract}$ in which $\overline{\Sigma} = (\overline{\sigma}^{(1)}, \cdots, \overline{\sigma}^{(\lambda)})$. Now, we renumber $\overline{\Sigma}$ as $(\sigma_1, \cdots, \sigma_{m\lambda})$ without changing their order. For the maximal identity $\mathsf{id} = (id_1, \cdots, id_\lambda)$, we build $\overline{\mathbf{h}}_{\mathsf{id}} = (\overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\lambda,id_\lambda)})$ and then compute $\mathsf{SK}_{\mathsf{id}}$ by calling $\mathsf{SampleTrap}$ for input $(\overline{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}_{\mathsf{id}}, \mathsf{MSK}, \overline{\Sigma})$. We now split $\overline{\mathbf{h}}_{\mathsf{id}}$ into $(\overline{\mathbf{h}}^{(1)}, \cdots, \overline{\mathbf{h}}^{(m\lambda)})$ and let $\overline{\mathbf{a}}^{(i)} = (\overline{\mathbf{a}}_\epsilon | \overline{\mathbf{h}}^{(1)} | \cdots | \overline{\mathbf{h}}^{(i)})$ with $\overline{\mathbf{a}}^{(0)} = \overline{\mathbf{a}}_\epsilon$. Then, $\mathsf{SampleTrap}$ calls $\mathsf{TrapDel}(\overline{\mathbf{a}}^{(i-1)}, \overline{\mathbf{h}}^{(i)}, \mathsf{td}^{(i-1)}, \sigma_i)$ up to $m\lambda$ times for $i \in [m\lambda]$, in which $\mathsf{td}^{(0)} = \mathsf{td}_\epsilon$ and $\mathsf{td}^{(i-1)}$ is the output of the previous execution of $\mathsf{TrapDel}(\overline{\mathbf{a}}^{(i-2)}, \overline{\mathbf{h}}^{(i-1)}, \mathsf{td}^{(i-2)}, \sigma_{i-1})$, for $2 \leq i \leq m\lambda$. Now, all σ_i's are set in the same way as in Section 3.3, that is, for $2 \leq i \leq m\lambda$, $\sigma_i \geq \omega(\sqrt{\log(d\gamma)}) \cdot \sqrt{7(s_1(\mathbf{R}^{(i-1)})^2 + 1)}$, and
$$s_1(\mathbf{R}^{(i-1)}) \leq \sqrt{((2d-1)t + (i-1)d\gamma\tau) \cdot (d\gamma\tau)} \cdot \omega(\log n) \cdot \sigma_{i-1},$$
in which $\mathbf{R}^{(i-1)}$ is the matrix representation, as in (13) with the last row's index $t+(i-1)\gamma\tau$, of the private key (the trapdoor) for $\overline{\mathbf{a}}^{(i-1)} = (\overline{\mathbf{a}}_\epsilon | \overline{\mathbf{h}}^{(1)} | \cdots | \overline{\mathbf{h}}^{(i-1)})$, with σ_1 and $\mathbf{R}^{(1)}$ play the role of σ and $\mathbf{T}$ in (7), (8).
- We set Gaussian parameters used in $\mathsf{Decrypt}$ $\overline{\Psi} = (\Psi_1, \cdots, \Psi_\lambda)$ as follows: For $\ell \in [\lambda]$, since Ψ_ℓ is used in $\mathsf{GenSamplePre}(\overline{\mathbf{f}}_{\mathsf{id}}, \mathsf{SK}_{\mathsf{id}}, u_0, \Psi_\ell)$ with $\mathsf{SK}_{\mathsf{id}} = \mathsf{td}_{\mathsf{id}}$ the trapdoor for $\overline{\mathbf{f}}_{\mathsf{id}} = (\overline{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$ which equals to $\overline{\mathbf{a}}^{(\ell m)}$ above. Therefore, for $\ell \in [\lambda - 1]$ we can set $\Psi_\ell = \sigma_{\ell m+1}$, and
$$\Psi_\lambda \geq \omega(\sqrt{\log(d\gamma)}) \cdot \sqrt{7(s_1(\mathbf{R}^{(m\lambda)})^2 + 1)},$$
$$s_1(\mathbf{R}^{(m\lambda)}) \leq \sqrt{((2d-1)t + (m\lambda)d\gamma\tau) \cdot (d\gamma\tau)} \cdot \omega(\log n) \cdot \sigma_{m\lambda},$$
in which $\mathbf{R}^{(m\lambda)}$ is the matrix representation for the private key (the trapdoor) for $\overline{\mathbf{a}}^{(m\lambda)} = (\overline{\mathbf{a}}_\epsilon | \overline{\mathbf{h}}^{(1)} | \cdots | \overline{\mathbf{h}}^{(m\lambda)})$.
- We set Gaussian parameters used in $\mathsf{Encrypt}$ $\overline{\alpha} = (\alpha_1, \cdots, \alpha_\lambda)$ such that for $\ell \in [\lambda]$, α_ℓ satisfies (16).

4.3 Security Analysis

Theorem 4. *The proposed* HIBE *system is INDr-sID-CPA secure in the standard model under the* DMPLWE *assumption.*

Proof. We construct a sequence of games from G_0 to G_4 in which an INDr–sID–CPA adversary can distinguish two consecutive games G_i and G_{i+1} *with negligible probability only.* In particular, for the transition of the last two games G_3 and

G_4, we show by contradiction that if there exists an adversary whose views are different in each game, i.e., the adversary can distinguish G_3 from G_4 with non-negligible probability, then we can build an adversary who can solve the underlying DMPLWE problem.

Game G_0 is the original IND–sID–CPA game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$. Note that, we are working with the selective game: at the beginning, $\mathcal{A}$ lets the challenger know the target identity $\mathsf{id}^* = (id_1^*, \cdots, id_\theta^*)$ that $\mathcal{A}$ intends to atack, where $\theta \leq \lambda$. Then, $\mathcal{C}$ runs Setup to choose randomly a vector of polynomials $\overline{\mathbf{a}}_\epsilon = (a_1, \cdots, a_{t'})$ together with an associated trapdoor td_ϵ, a set of polynomial vectors sampled randomly $\overline{\mathbf{h}}^{(i,\mathsf{bit})} = (h_1^{(i,\mathsf{bit})}, \cdots, h_{t'}^{(i,\mathsf{bit})})$, which are stored in $\mathsf{HList0}$, where each $h_j^{(i,\mathsf{bit})} \in \mathbb{Z}_q^{<n}[x]$ for $j \in [t]$, and each $h_j^{(i,\mathsf{bit})} \in \mathbb{Z}_q^{<n+d-1}[x]$ for $j \in \{t+1, \cdots, t+\gamma\tau\}$, and $\mathcal{C}$ also chooses a random polynomial $u_0 \in \mathbb{Z}_q^{<n+2d-2}[x]$. The challenger then sets $\mathsf{MSK} := \mathsf{td}_\epsilon$ as the master secret key. Furthermore, at the Challenge Phase, the challenger also generates a challenge ciphertext $\overline{\mathsf{CT}}^*$ for the identity id^*.

Game G_1 is the same as G_0 except that in the Setup Phase the challenger $\mathcal{C}$ generates $(\overline{\mathbf{h}}^{(i,\mathsf{bit})})_{0 \leq i \leq \lambda, \mathsf{bit} \in \{0,1\}}$ stored in $\mathsf{HList1} := \{(i, \mathsf{bit}, \overline{\mathbf{h}}^{(i,\mathsf{bit})}) : i \in [\lambda], \mathsf{bit} \in \{0,1\}\}$ with the corresponding trapdoor $\mathsf{td}^{(i,\mathsf{bit})}$ stored in $\mathsf{TList1} := \{(i, \mathsf{bit}, \mathsf{td}^{(i,\mathsf{bit})}) : i \in [\lambda], \mathsf{bit} \in \{0,1\}\}$ using $\mathsf{TrapGen}$.

Game G_2 is the same as G_1, except that the challenger $\mathcal{C}$ does not use td_ϵ as the master secret key nor the $\mathsf{Extract}$ algorithm to response a private key queries on $\mathsf{id} = (id_1, \cdots, id_\ell)$ which is not a prefix of the target id^*, where $\ell \leq \lambda$. Instead, $\mathcal{C}$ designs a new procedure $\mathsf{TrapExtract}$ with the knowledge of $\mathsf{TList1}$. $\mathsf{TrapExtract}$ requires not all of $\mathsf{TList1}$ but only one $\mathsf{td}^{(j,id_j)} \in \mathsf{TList1}$ for any $j \in [\ell]$.

$\mathsf{TrapExtract}(\overline{\mathbf{a}}_\epsilon, \mathsf{HList1}, \mathsf{id} = (id_1, \cdots, id_\ell), j, \mathsf{td}^{(j,id_j)})$:

1. Build $\overline{\mathbf{f}}_{\mathsf{id}} = (\overline{\mathbf{a}}_\epsilon, \overline{\mathbf{h}}^{(1,id_1)}, \cdots, \overline{\mathbf{h}}^{(j-1,id_{j-1})}, \overline{\mathbf{h}}^{(j+1,id_{j+1})}, \cdots, \overline{\mathbf{h}}^{(\ell,id_\ell)})$
2. $\mathsf{SK}_{\mathsf{id}} \leftarrow \mathsf{SampleTrap}(\overline{\mathbf{h}}^{(j,id_j)}, \overline{\mathbf{f}}_{\mathsf{id}}, \mathsf{td}^{(j,id_j)}, \overline{\Sigma}^{(\ell)})$

Game G_3 is the same as G_2, except that in the Setup Phase, the challenger $\mathcal{C}$ generates $\mathsf{HList3}$ as follows:

- For each $j \in [\lambda]$ and $\mathsf{bit} \in \{0,1\}$ such that $\mathsf{bit} \neq id_j^*$, $\mathcal{C}$ calls $\mathsf{TrapGen}$ to generate $\overline{\mathbf{h}}^{(j,\mathsf{bit})}$ and its associated trapdoor $\mathsf{td}^{(j,\mathsf{bit})}$.
- For each $j \in [\lambda]$ and $\mathsf{bit} \in \{0,1\}$ such that $\mathsf{bit} = id_j^*$, $\mathcal{C}$ simply samples $\overline{\mathbf{h}}^{(j,\mathsf{bit})}$ uniformly at random and set $\mathsf{td}^{(j,\mathsf{bit})} = \perp$.

The challenger then put all $\overline{\mathbf{h}}^{(j,\mathsf{bit})}$ into $\mathsf{HList3}$ and all $\mathsf{td}^{(j,\mathsf{bit})}$ into $\mathsf{TList3}$. At the moment, to response a private key query on identity $\mathsf{id} = (id_1, \cdots, id_\ell)$ which is not a prefix of the target identity id^*, the challenger chooses an index $j^\dagger$ such that

$id_{j^\dagger} \neq id^*_{j^\dagger}$. It then runs $\mathsf{TrapExtract}(\overline{\mathbf{a}}_\epsilon, \mathsf{HList3}, \mathsf{id} = (id_1, \cdots, id_\ell), j^\dagger, \mathsf{td}^{(j^\dagger, id_{j^\dagger})})$, where $\mathsf{td}^{(j^\dagger, id_{j^\dagger})} \in \mathsf{TList3}$, and gives the result $\mathsf{SK}_{\mathsf{id}}$ to the adversary. At the Challenge Phase, the challenge ciphertext $\overline{\mathsf{CT}}^*$ is generated by computing $\mathsf{Encrypt}(\mathsf{id}, \mu, u_0, \overline{\alpha})$ over $\mathsf{HList3}$.

<u>**Game** G_4</u> is the same as G_3, except that the challenge ciphertext $\overline{\mathsf{CT}}^* = (\mathsf{CT}^*_0, \mathsf{CT}^*_1)$ is chosen uniformly at random by the challenger.

In what follows, we show the indistinguishability of the games. It is easy to see that the view of the adversary is identical in games G_0 and G_1, in games G_1 and G_2, in games G_2 and G_3, except in games G_3 and G_4. We show that the view of the adversary is indistinguishable in these two games. We proceed by contradiction. Assume that the adversary $\mathcal{A}$ can distinguish between games G_3 and G_4 with non-negligile probability. Then we construct an adversary $\mathcal{B}$ that is able to solve DMPLWE problem with the same probability. The reduction from DMPLWE is as follows:

- **Instance:** Assume that the goal of $\mathcal{B}$ is to decide whether $1+t'(\ell+1)$ samples (f_z, ct_z) for $z \in \{0, \cdots, t'(\ell+1)\}$ (i) follow $\prod_{z=0}^{t'(\ell+1)} \mathcal{U}(\mathbb{Z}_q^{n'-d_z}[x] \times \mathbb{R}_q^{d_z[x]})$, or (ii) follow $\mathsf{DMP}_{q,n',\mathbf{d},\chi}(s)$, where $n' = n + 2d + k$ and
  - $\mathbf{d} = (d_0, d_1, \cdots, d_{t'(\ell+1)})$ is interpreted as follows: $d_0 := k+2$ and for $i \in \{0, \cdots, \ell\}$, $d_{i \cdot t' + j} = \begin{cases} 2d + k, & \text{if } j \in [t], \\ d + k + 1, & \text{if } j \in \{t+1, \cdots, t+\gamma\tau\}. \end{cases}$
  - f_z are random in $\mathbb{Z}_q^{<n'-d_z}[x]$ for $z \in \{0, \cdots, t'(\ell+1)\}$.

  In other words, $\mathcal{B}$ has to distinguish whether (i) all ct_z are random or (ii) $\mathsf{ct}_z = f_z \odot_{d_z} s + 2e_z$ in $\mathbb{Z}_q^{<d_z}[x]$, for some $s \xleftarrow{\$} \mathbb{Z}_q^{<n'-1}[x]$ and $e_z \leftarrow \chi^{d_z}[x]$, for all $z \in \{0, \cdots, t'(\ell+1)\}$.
- **Targeting:** $\mathcal{B}$ receives from the adversary $\mathcal{A}$ the target identity id^* that $\mathcal{A}$ wants to attack.
- **Setup:**. $\mathcal{B}$ generates HListB in the same way as in Game G_3 and Game G_4 as follows:
  - For each $j \in [\lambda]$ and $\mathsf{bit} \in \{0,1\}$ such that $\mathsf{bit} \neq id^*_j$: $\mathcal{B}$ calls $\mathsf{TrapGen}$ to generate $\overline{\mathbf{h}}^{(j,\mathsf{bit})}$ and its associated trapdoor $\mathsf{td}^{(j,\mathsf{bit})}$.
  - For each $j \in [\lambda]$ and $\mathsf{bit} \in \{0,1\}$ such that $\mathsf{bit} = id^*_j$: $\mathcal{B}$ simply samples $\overline{\mathbf{h}}^{(j,\mathsf{bit})}$ uniformly at random and set $\mathsf{td}^{(j,\mathsf{bit})} = \perp$.

  The challenger then put all $\overline{\mathbf{h}}^{(j,\mathsf{bit})}$ into HListB and all $\mathsf{td}^{(j,\mathsf{bit})}$ into TListB.
- **Queries**: To response the private key queries, $\mathcal{B}$ acts as in Game G_3 or in Game G_4 using one of trapdoors that is not $\perp$.
- **Challenge:** To produce the challenge ciphertext, $\mathcal{B}$ chooses randomly $b \xleftarrow{\$} \{0,1\}$ and sets $\overline{\mathsf{CT}}^* := (\mathsf{CT}^*_0 := ct_0 + \mu, \mathsf{CT}^*_1 := (\mathsf{ct}_1, \cdots, \mathsf{ct}_{t'(\ell+1)}))$.
- **Guess:** Eventually, $\mathcal{A}$ has to guess and output the value of b. Then, $\mathcal{B}$ returns what $\mathcal{A}$ outputted.

Analysis. Clearly, from the view of $\mathcal{A}$, the behaviour of $\mathcal{B}$ is almost identical in both Games G_3 and G_4. The only different thing is producing the challenge ciphertext. Specifically, if ct_z's are DMPLWE samples then the components of $\overline{\mathsf{CT}}^*$ are distributed as in Game G_3, while ct_z's are random then the components of $\overline{\mathsf{CT}}^*$ are distributed as in Game G_4. Since $\mathcal{A}$ can distinguish between Games G_3 and G_4 with non-negligile probability, then so can $\mathcal{B}$ in solving DMPLWE with the same probability. □

5 Conclusions

In this paper, we present a trapdoor delegation method that enables us to obtain a trapdoor for an expanded set of polynomials from a given trapdoor for a subset of the set. Also, thanks to the polynomial trapdoor delegation, we built a hierarchical identity–based encryption system that is secure in the standard model under the DMPLWE assumption.

Acknowledgment. We all would like to thank anonymous reviewers for their helpful comments. This work is partially supported by the Australian Research Council Discovery Project DP200100144. The first author has been sponsored by a Data61 PhD Scholarship. The fourth author has been supported by the Australian ARC grant DP180102199 and Polish NCN grant 2018/31/B/ST6/03003.

References

1. Agrawal, S., Boneh, D.: Identity-Based Encryption from Lattices in the Standard Model. In: Manuscript (2009), http://www.robotics.stanford.edu/~xb/ab09/latticeibe.pdf
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 553–572. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
3. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. pp. 255–271. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/10.1007/3-540-39200-9_16
4. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai Trees, or How to Delegate a Lattice Basis. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. pp. 523–552. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
5. Dodis, Y., Fazio, N.: Public Key Broadcast Encryption for Stateless Receivers. In: Feigenbaum, J. (ed.) Digital Rights Management. pp. 61–80. Springer Berlin Heidelberg, Berlin, Heidelberg (2003). https://doi.org/0.1007/978-3-540-44993-5_5
6. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1374376.1374407
7. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) Advances in Cryptology — ASIACRYPT 2002. pp. 548–566. Springer Berlin Heidelberg, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-2_34

8. Hengartner, U., Steenkiste, P.: Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information. In: First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05). pp. 384–396 (2005). https://doi.org/10.1109/SECURECOMM.2005.18
9. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) Advances in Cryptology — EUROCRYPT 2002. pp. 466–481. Springer Berlin Heidelberg, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-46035-7_31
10. Lombardi, A., Vaikuntanathan, V., Vuong, T.D.: Lattice Trapdoors and IBE from Middle-Product LWE. In: Hofheinz, D., Rosen, A. (eds.) Theory of Cryptography. pp. 24–54. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-36030-6_2
11. Lyubashevsky, V., Micciancio, D.: Generalized Compact Knapsacks Are Collision Resistant. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) Automata, Languages and Programming. pp. 144–155. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11787006_13
12. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. pp. 1–23 (2010). https://doi.org/10.1007/978-3-642-13190-5_1
13. Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques. pp. 700–718. EUROCRYPT'12, Springer-Verlag, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
14. Micciancio, D., Regev, O.: Worst-Case to Average-Case Reductions Based on Gaussian Measures. SIAM J. Comput. **37**(1), 267–302 (Apr 2007). https://doi.org/10.1137/S0097539705447360
15. Pan, V.Y.: Structured Matrices and Polynomials: Unified Superfast Algorithms. Springer-Verlag, Berlin, Heidelberg (2001)
16. Roşca, M., Sakzad, A., Stehlé, D., Steinfeld, R.: Middle-Product Learning with Errors. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017. pp. 283–297. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-63697-9_10
17. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology. pp. 47–53. Springer Berlin Heidelberg, Berlin, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5
18. Stehlé, D., Steinfeld, R., Tanaka, K., Xagawa, K.: Efficient public key encryption based on ideal lattices. In: Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings. pp. 617–635 (2009). https://doi.org/10.1007/978-3-642-10366-7_36
19. Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In: Proceedings of the 11th ACM Conference on Computer and Communications Security. p. 354–363. CCS '04, Association for Computing Machinery, New York, NY, USA (2004). https://doi.org/10.1145/1030083.1030130