이학 박사 학위논문

A study on ID-based Homomorphic Encryption with Noisy Key

(잡음키를 가지는 신원기반 동형암호에 관한 연구)

2020년 2월

서울대학교 대학원

수리과학부

손용하

A study on ID-based Homomorphic Encryption with Noisy Key

(잡음키를 가지는 신원기반 동형암호에 관한 연구)

지도교수 천정희

이 논문을 이학 박사 학위논문으로 제출함

2019년 10월

서울대학교 대학원

수리과학부

손용하

손용하의 이학 박사 학위논문을 인준함

2019년 12월

위 원 장	김 명 환	(인)
부 위 원 장	천 정 희	(인)
위 원	현 동 훈	(인)
위 원	이 향 숙	(인)
위 원	윤 아 람	(인)

A study on ID-based Homomorphic Encryption with Noisy Key

A dissertation
submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
to the faculty of the Graduate School of
Seoul National University

by

Yongha Son
Dissertation Director : Professor Jung Hee Cheon

Department of Mathematical Sciences
Seoul National University

February 2020

Abstract

A study on ID-based Homomorphic Encryption with Noisy Key

Yongha Son

Department of Mathematical Sciences

The Graduate School

Seoul National University

Secure data analysis delegation on cloud is one of the most powerful application that homomorphic encryption (HE) can bring. As the technical level of HE arrive at practical regime, this model is also being considered to be a more serious and realistic paradigm. In this regard, this increasing attention requires more versatile and secure model to deal with much complicated real world problems.

First, as real world modeling involves a number of data owners and clients, an authorized control to data access is still required even for HE scenario. Second, we note that although homomorphic operation requires no secret key, the decryption requires the secret key. That is, the secret key management concern still remains even for HE. Last, in a rather fundamental view, we thoroughly analyze the concrete hardness of the base problem of HE, so-called Learning With Errors (LWE). In fact, for the sake of efficiency, HE exploits a weaker variant of LWE whose security is believed not fully understood.

For the data encryption phase efficiency, we improve the previously suggested NTRU-lattice ID-based encryption by generalizing the NTRU concept into module-NTRU lattice. Moreover, we design a novel method that decrypts the resulting ciphertext with a noisy key. This enables the decryptor to use its own noisy source, in particular biometric, and hence fundamentally solves the key management problem. Finally, by considering further improvement on existing LWE solving algorithms, we propose new algorithms that shows much faster performance. Consequently, we argue that the HE parameter choice should be updated regarding our attacks in order to maintain the currently claimed security level.

Key words: ID-based cryptography, Post-quantum cryptography, Homomorphic encryption, Noisy key cryptography
Student Number: 2014-21208

Contents

Chapter 1

Introduction

Homomorphic encryption (HE) is one of the most fascinate modern cryptographic primitives, which allows computations on encrypted state without secret key. This fundamentally removes the possibility of data leakage by storing only ciphertexts, and newly opens various applications that was impossible before the advent of HE. After the first proposal of HE by Gentry [Gen09], there has reported numerous contributions on functionality and efficiency of HE over a decade, and now its technical level is considered to be reach a quite practical extent. In accordance with this development, there has been reported some series of researches of secure data analysis on cloud. In this scenario, data owners encrypt their data, and those encrypted data gather into cloud where data analysis would be homomorphically done. So far, this scenario is examined in somewhat naive sense where encryption and decryption is done by a sole data owner, or there is only one massive amount of data owner. However, the actual real world problem is likely to be much more complicated than such model, and hence a more delicate argument on key distribution and data access management

is required.

1.1 Access Control based on Identity

Consider a data analysis model that consists of several data owners that provides each own data in encrypted state and clients that query analysis result of data. Then, it is highly desirable for each data owner to have a control on authority of access for its data, in a point that this is connected to business model. The most natural solution for this would be using ID-based HE, where each data owner encrypts its data so that only the target ID user can decrypt it. It is actually realized by a generic compiler which converts a plain ID-based encryption into ID-based HE is reported in [GSW13]. Hence our goal is achieved by letting each data owner encrypts its data with regard to every ID that it wants to assign access.

However, this solution still has efficiency issues. First in this case, the computation cost of each data owner would be proportional to the number of ID, since it should encrypt data for each other ID. As the ciphertext expansion rate of homomorphic encryption is rather small, this would be a burden for data owners who are not expected to have huge computation power. In this regard, it would be greatly helpful to consider the approach similar to [GHS12] that studied homomorphic evaluation of AES circuit, which converts AES ciphertext into HE ciphertext. Then secondly, for homomorphic evaluation of ID-based encryption, now the efficiency of base ID-based encryption matters. For this, there is one quite efficient lattice-based scheme [DLP14] is proposed base on NTRU problem, whose efficiency is believed to be comparable to a recent implementation of pairing-based one [BF01].

1.2 Biometric Key Management

Although homomorphic operations between ciphertexts can be done without secret key, we cannot still exclude the secret key in the whole scenario, at least for decryption procedure. In other words, there still remains key management problem. Fuzzy extractor, suggested by Dodis *et al.* [DORS08], is a promising cryptographic primitive that resolves those problems. Informally, fuzzy extractor extracts a uniform random string $\boldsymbol{r}$ and a public value H called helper from a reading $\boldsymbol{w}$ in a random source. Then with the helper H and another reading $\boldsymbol{w}'$ of the same source, one can reproduce the same random string whenever $\boldsymbol{w}$ and $\boldsymbol{w}'$ are close; which means, in biometrics setting, $\boldsymbol{w}$ and $\boldsymbol{w}'$ come from the same person.

There have been several proposals of fuzzy extractor, and most of them rely on another cryptographic primitive named *secure sketch* also suggested by the seminal work of Dodis *et al.* [DORS08]. Until now, many fuzzy extractors using secure sketch is being developed [WL18, WLG19] and it currently tolerates considerably high amount of error—linear fraction of errors— in polynomial time on standard assumption. However, the most critical weakness of secure sketch-based constructions is its too high entropy requirement for random source. That is, known building techniques of secure sketch requires too high min-entropy of random sources, and it is still difficult to obtain such random sources in practice [Dau09, KLRW14].

1.3 Concrete Security of HE

The semantic security of HE is based on the hardness of lattice-problem named Learning With Errors (LWE). However, for the sake of efficiency, most of HE implementations uses extremely small vectors, which currently

lie outside of the currently known provably secure parameter regime. In this situation, Albrecht [Alb17] recently pointed out that the variants of LWE with small key is far weaker than previous thoughts by suggesting a new variant of the *dual attack*, which is one of primary solving algorithms for LWE. About this issue, in the *homomorphic encryption standardization* [ACC+18], HE community reaches a consensus of using *ternary* secrets while expecting there would be no more significant improvement on ternary secrets. However for the use of *sparse* secrets, it represents some uncertainty by stating

"However, we will not present tables for sparse secrets because the security implications of using such sparse secrets is not well understood yet."

1.4 List of Papers

This thesis contains the results of the following papers.

- [CHHS19] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son, A Hybrid of Dual and Meet-in-the-Middle Attack on Sparse and Ternary Secret LWE, *IEEE Access, Vol. 7, 2019.*
- [SC19] Yongha Son, and Jung Hee Cheon. Revisiting the Hybrid attack on sparse secret LWE and Application to HE parameters, *7th Workshop on Encrypted Computing & Applied Homomorphic Cryptography (WAHC),* 2019.
- [CHS19] Jung Hee Cheon, Minki Hhan, and Yongha Son, Reusable Fuzzy Extractors from Local Functions, In submission.

- [CKKS19] Jung Hee Cheon, Duhyeong Kim, Taechan Kim, and Yongha Son, A New Trapdoor over Module-NTRU Lattice and its Application to ID-based Encryption, In submission.

Chapter 2

Background

2.1 Notation

We first denote $[n] = \{0, 1, \cdots, n-1\}$, and $\mathbb{Z}_n$ is treated as $[n]$ in this thesis. For $a \in \mathbb{Z}$, we denote $a \bmod n$ by a unique number $\in [0, n)$ such that $a - (a \bmod p)$ is an integer multiple of n. $\lfloor a \rceil$ denotes the nearest integer of a, and $[a]_p$ is a unique integer in $(-p/2, p/2]$ such that $a - [a]_p$ is a multiple of p.

Column vectors are written by bold and lower case letters and matrices are written by upper case letters. The entries of bold face is denoted as $\boldsymbol{v} = (v_0, v_1, \cdots, v_{n-1})^t$. We sometimes take modular n for indices of vector and omit the transpose operator t.

2.2 Lattices

A lattice is a discrete additive subgroup of $\mathbb{R}^d$. A full rank matrix $B \in \mathbb{R}^{d\times n}$ is called a basis of a lattice Λ if it holds that

$$\Lambda = \{B\boldsymbol{x} : \boldsymbol{x} \in \mathbb{Z}^n\}.$$

We write $\Lambda(B)$ to represent a lattice determined by basis B. The dimension of a lattice Λ is defined as the cardinality of any basis of Λ. In particular, a lattice in $\mathbb{R}^d$ whose dimension is maximal is called full-rank lattice and without any special mention, we will only consider full-rank lattices throughout this paper.

The fundamental parallelepiped of a lattice basis $B = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_d] \in \mathbb{R}^{d\times d}$ is given by

$$\mathcal{P}(B) = \left\{ \boldsymbol{x} \in \mathbb{R}^d \mid \boldsymbol{x} = \sum_{i=1}^{d} c_i \boldsymbol{b}_i \text{ for } -1/2 \leqslant c_i < 1/2 \right\}$$

The determinant det of lattice Λ is defined as the d-dimensional volume of its fundamental parallelepiped.

2.2.1 Lattice Reduction Algorithm

Lattice reduction algorithm with root-Hermite factor δ_0 returns a short basis, especially whose first vector $\boldsymbol{b}_1$ has size $\leqslant \delta_0^d \cdot \det \Lambda^{1/d}$. The BKZ algorithm [CN11] is a commonly used lattice reduction algorithm. For inputs d-dimensional basis B of some lattice and *blocksize* β, the BKZ algorithm repeatedly solves the shortest vector problem (SVP) on dimension β blocks obtained from B, and it is known that BKZ terminates after polynomial

numbers of SVP solver call. Thus the time complexity of BKZ closely related to the core SVP oracle call, and we will mention the explicit formula in later Section 5.5. We denote an BKZ algorithm call with blocksize β for a basis T by $\mathsf{BKZ}_\beta(T)$.

Regarding the quality of BKZ algorithm, in [Che13] it is experimentally verified that BKZ with blocksize β yields root-Hermite factor

$$\delta_0 \approx \left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}},$$

and we also accept this for our analysis.

2.2.2 BKZ cost model

There are two popular choices for BKZ cost model according to core SVP solver; one is from a sieving algorithm [BDGL16] and the other from an enumeration algorithm [CN11]. For blocksize β and dimension d, we assume $T_{BKZ}(\beta, d)$ costs by

- $8d \cdot 2^{0.292\beta+16.4}$ according to sieving,
- $8d \cdot 2^{0.187\beta\log\beta-1.019\beta+16.1}$ according to enumeration.

2.2.3 Geometric Series Assumption (GSA)

There is an useful assumption that estimates the lengths of the Gram-Schmidt vectors of a reduced basis. Let $B \in \mathbb{Z}^{d\times d}$ be a reduced basis of some full-ranked lattice with root-Hermite factor δ_0 and let $\boldsymbol{b}_i^*$ denote the i-th Gram-Schmidt vectors of B. Then the geometric series assumption (GSA) predicts that the length of $\boldsymbol{b}_i^*$ decreases geometrically. More precisely, GSA

predicts $R_i := \|\boldsymbol{b}_i^*\|$ by

$$R_i = \delta_0^{-2(i-1)+d} \cdot \det(\Lambda(B))^{1/d}. \tag{2.1}$$

2.2.4 The Nearest Plane Algorithm

We will exploit Babai's nearest plane algorithm [Bab86] (denoted by NP shorthand) in our attack as a subroutine, whose property is summarized as following.

Lemma 2.2.1. *Let B be a lattice basis and $\boldsymbol{t} \in \mathbb{R}^d$ be a target vector. Then Babai's nearest plane algorithm NP given input B and $\boldsymbol{t}$ returns the unique vector $\boldsymbol{e} = \mathsf{NP}_B(\boldsymbol{t}) \in \mathcal{P}(B^*)$ satisfying $\boldsymbol{t} - \boldsymbol{e} \in \Lambda(B)$, where B^* is the Gram-Schmidt basis of B.*

We denote the output vector by $\mathsf{NP}_B(\boldsymbol{t}) = \boldsymbol{e}$. For the runtime of nearest plane algorithm, we follow the heuristic assumption due to Hirschhorn et al. [HHHGW09], which says the number of operations T_{NP} of NP algorithm on d-dimensional lattice input is upper bounded by

$$T_{\mathsf{NP}} = d^2/2^{1.06}. \tag{2.2}$$

For more details on the nearest plane algorithm, we refer Babai's original work [Bab86] or Linder and Peikert's work [LP11].

2.3 Gaussian Measures

For a full-rank n-dimensional lattice $\Lambda \subset \mathbb{R}^n$, *the discrete Gaussian distribution* with width $\sigma > 0$ and center $\boldsymbol{c} \in \mathbb{R}^n$ denoted by $\mathcal{D}_{\Lambda,\sigma,\mathbf{c}}$ is a

distribution over Λ which samples $\boldsymbol{x} \in \Lambda$ with the probability

$$\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}(\boldsymbol{x}) := \frac{\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{x})}{\sum_{\boldsymbol{z}\in\Lambda}\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{z})}$$

where $\rho_{\sigma,\boldsymbol{c}}(\boldsymbol{z}) := \exp\left(-\frac{\|\boldsymbol{z}-\boldsymbol{c}\|^2}{2\sigma^2}\right)$.

There is an well-known parameter $\eta_\varepsilon(\Lambda)$ called *smoothing parameter* defined by [MR07], which is defined by the smallest $s > 0$ such that

$$\rho_{1/s,\mathbf{0}}(\Lambda^*\backslash\{\mathbf{0}\}) \leqslant \varepsilon,$$

where Λ^* is a dual lattice of Λ. We also denote the scaled-version $\eta'_\varepsilon(\Lambda) := \frac{1}{\sqrt{2\pi}}\eta_\varepsilon(\Lambda)$. In particular, it is known that from [GPV08]

$$\eta'_\varepsilon(\mathbb{Z}) \approx \frac{1}{\pi}\cdot\sqrt{\frac{1}{2}\ln\left(2+\frac{2}{\varepsilon}\right)}.$$

A Gaussian Sampler.

An algorithm that approximately samples the discrete Gaussian is proposed by [GPV08], and we will use for our MNTRU lattices and IBE scheme. Here we omit the detail of the algorithm and simply define the syntax: for a basis $\mathbf{B}$ of a lattice L, we denote the [GPV08] algorithm that approximately samples $\mathcal{D}_{\Lambda,\sigma,\boldsymbol{c}}$ by

$$\mathsf{GaussianSampler}(\mathbf{B},\sigma,\boldsymbol{c}).$$

2.3.1 Kullback-Leibler Divergence

Instead of the traditional statistical distance concept to measure the distance of two distributions, we especially will make use of Kullback-Leibler divergence (or KL divergence) following the methodology of [DLP14].

Remark. In fact, the recent literature is using more general concept of distance called *Rényi divergence*, for example in [PAFZ19]. However, the previous work [DLP14] was analyzed with KL divergence, and hence in this paper we stick to the KL divergence for a clear comparison.

Definition 2.3.1 (Kullback-Leibler Divergence)**.** *Let $\mathcal{P}$ and $\mathcal{Q}$ be two distributions over a common countable set Ω, and let $S \subset \Omega$ be the support of $\mathcal{P}$. The Kullback-Leibler Divergence, noted D_{KL} of $\mathcal{Q}$ from $\mathcal{P}$ is defined as:*

$$D_{KL}(\mathcal{P}||\mathcal{Q}) = \sum_{i \in S} \ln\left(\frac{\mathcal{P}(i)}{\mathcal{Q}(i)}\right)\mathcal{P}(i)$$

with the convention that $\ln(x/0) = +\infty$ for any $x > 0$.

It is known that, if two distribution $\mathcal{P}$ and $\mathcal{Q}$ has small KL divergence, hardness of any search problem that requires oracle queries for $\mathcal{P}$ is preserved even if the oracle queries is replaced with $\mathcal{Q}$.

Lemma 2.3.1 (Lemma 1 of [PDG14])**.** *Let $\mathcal{A}^{\mathcal{P}}$ be an algorithm making at most q queries to an oracle sampling from a distribution $\mathcal{P}$ and returning a bit. Let $\mathcal{A} \geqslant 0$, and $\mathcal{Q}$ be a distribution such that $D_{KL}(\mathcal{P}||\mathcal{Q}) \leqslant \varepsilon$. Let x (resp. y) denote the probability that $\mathcal{A}^{\mathcal{P}}$ (resp. $\mathcal{A}^{\mathcal{Q}}$) outputs 1. Then,*

$$|x - y| \leqslant \sqrt{\frac{q\varepsilon}{2}}.$$

Finally, we have the following fact for KL divergence of the ideal discrete Gaussian and the Gaussian sampler that we will use.

Theorem 2.3.1 (Theorem 2 of [DLP14]). *For any $\varepsilon \in (0, 1/4n)$, if $\sigma \geqslant \eta'_\varepsilon(\mathbb{Z})\|\boldsymbol{B}^*\|$, then*

$$D_{KL}\left(\mathcal{D}_{\Lambda(\boldsymbol{B}),\sigma,\boldsymbol{c}} || \mathsf{GaussianSampler}(\boldsymbol{B},\sigma,\boldsymbol{c})\right) \leqslant 2\left(1-\left(\frac{1+\varepsilon}{1-\varepsilon}\right)^n\right)^2 \approx 8n^2\varepsilon^2.$$

2.4 Lattice-based Hard Problems

2.4.1 The Learning With Errors Problem

Let $n, q > 0$ be integers, $\boldsymbol{s} \in \mathbb{Z}_q^n$ and χ be an error distribution over $\mathbb{Z}$. We define a distribution $\mathcal{A}^{LWE}_{n,q,\chi,\boldsymbol{s}}$ over $\mathbb{Z}_q^{n+1}$ obtained by sampling $\boldsymbol{a} \leftarrow \mathcal{U}(\mathbb{Z}_q^n)$ and $e \leftarrow \chi$, and then computing

$$(\boldsymbol{a}, b) = (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e) \in \mathbb{Z}_q^{n+1}.$$

Given many samples $(\boldsymbol{a}_i, b_i)$ from $\mathcal{A}^{LWE}_{n,q,\chi,\boldsymbol{s}}$, we can represent it by a matrix $(A, \boldsymbol{b})$ whose each row corresponds to one sample, and denoted it by LWE samples. Also we define $\mathcal{A}^{LWE}_{n,q,\alpha,\boldsymbol{s}}$ as the distribution $\mathcal{A}^{LWE}_{n,q,\chi,\boldsymbol{s}}$ where χ is a Gaussian distribution $\mathcal{D}_{\mathbb{Z},\alpha q}$ for $\alpha > 0$.

Definition 2.4.1 (Learning with Errors). *Let $\mathcal{S}$ be a distribution over $\mathbb{Z}_q^n$.*

- *A search version of $\mathsf{LWE}_{n,q,\chi}(\mathcal{S})$ (or $\mathsf{LWE}_{n,q,\alpha}(\mathcal{S})$) is a problem that asks to find the secret key $\boldsymbol{s}$, given LWE samples from $\mathcal{A}^{LWE}_{n,q,\chi,\boldsymbol{s}}$ (or $\mathcal{A}^{LWE}_{n,q,\alpha,\boldsymbol{s}}$) for a fixed $\boldsymbol{s} \leftarrow \mathcal{S}$.*

- *A decision version of $\mathsf{LWE}_{n,q,\chi}(\mathcal{S})$ (or $\mathsf{LWE}_{n,q,\alpha}(\mathcal{S})$) is a problem that asks to determine that, given arbitrarily many samples $(\boldsymbol{a}_i, b_i) \in \mathbb{Z}_q^{n+1}$, they are LWE samples from $\mathcal{A}^{LWE}_{n,q,\chi,\boldsymbol{s}}$ (or $\mathcal{A}^{LWE}_{n,q,\alpha,\boldsymbol{s}}$) for a fixed $\boldsymbol{s} \leftarrow \mathcal{S}$ or uniform random samples from $\mathcal{U}(\mathbb{Z}_q^{n+1})$.*

Although we abuse the notation LWE for both search and decision problem, without special mention, we consider a decision version of LWE problem for the most cases in this thesis. Also note that there is a decision-to-search reduction of LWE problem [Reg05].

Special Distributions for Secret Vectors.

Several LWE-based cryptosystems takes the secret distribution $\mathcal{S}$ by small portion of $\mathbb{Z}_q^n$ to enhance efficiency. In particular, we will focus on the case where $\mathcal{S}$ is the set of sparse (signed) binary vectors. For the sake of readability, we denote

$$\mathcal{B}_{n,h} = \{\boldsymbol{s} \in \{\pm 1, 0\}^n : \mathsf{HW}(\boldsymbol{s}) = h\},$$
$$\mathcal{B}_{n,\leqslant h} = \{\boldsymbol{s} \in \{\pm 1, 0\}^n : \mathsf{HW}(\boldsymbol{s}) \leqslant h\}.$$

2.4.2 NTRU Problem

We recall the definition of the NTRU lattices.

Definition 2.4.2 (NTRU lattices)**.** *Let n be a power-of-two integer, and q be a positive integer. For $f, g \in \mathcal{R}$, let $h = g/f \mod q$. The NTRU lattice Λ_{NTRU} associated to h and q is*

$$\Lambda_{NTRU} = \{(u, v) \in \mathcal{R}^2 : u + vh = 0 \mod q\}.$$

By the definition, Λ_{NTRU} can also be seen as a full-rank lattice in $\mathbb{Z}^{2n}$ generated by the columns of $\mathbf{A}_{\mathrm{NTRU}} = \begin{pmatrix} -\mathcal{A}_n(h) & qI_n \\ I_n & O_n \end{pmatrix}$.

Several cryptosystems that deal with the NTRU lattices base their security on the hardness assumption of the NTRU problem which states that

if $f, g \in R_q$ are random small polynomials, their quotient g/f is indistinguishable from random in R_q.

An interesting aspect of the NTRU lattice is that it can be easily instantiated with a trapdoor basis. More precisely, as explained in [HHGP+03], one can find another basis by computing $F, G \in \mathcal{R}$ such that $gF - fG = q$, and then a short trapdoor basis of Λ_{NTRU} is provided by the integral matrix

$$\mathbf{T}_{\mathrm{NTRU}} := \begin{pmatrix} \mathcal{A}_n(g) & \mathcal{A}_n(G) \\ -\mathcal{A}_n(f) & -\mathcal{A}_n(F) \end{pmatrix}.$$

2.5 One-way and Pseudo-random Functions

Let $F : \{0,1\}^n \to \{0,1\}^m$ be a function. Usually, one-way functions (OWF) and pseudo-random generators (PRG) are defined for uniformly chosen $\boldsymbol{x} \leftarrow \mathcal{U}_n$. We here consider more general definitions where the input distribution can be *nonuniform*, and the advantages are also relaxed.

Definition 2.5.1 (OWF over weak seed)**.** *Let $\mathcal{W} = \{W_n\}$ be a family of distributions over $\{0,1\}^n$, and let $F = \{F_n : \{0,1\}^n \to \{0,1\}^{n^s}\}$ be a family of functions. For a PPT adversary $\mathcal{A}$, we define*

$$\mathsf{Adv}_{G_n,W_n}(\mathcal{A}) := \Pr_{\mathcal{A}, w \xleftarrow{\$} W_n}[F_n(w') = F(w) : \mathcal{A}(F_n, F_n(w)) \to w'].$$

F is called to be $\varepsilon(n)$-one-way (or ε-OW for short) over $\mathcal{W}$ if for any PPT adversary $\mathcal{A}$, it holds that

$$\mathsf{Adv}_{F_n,W_n}(\mathcal{A}) \leqslant \varepsilon(n).$$

In particular, if $\varepsilon = \mathsf{negl}(n)$, we say that F is a one-way function over

$\mathcal{W}$.

Definition 2.5.2 (PRG over weak seed)**.** *Let $\mathcal{W} = \{W_n\}$ be a family of distributions over $\{0,1\}^n$, and let $F = \{F_n : \{0,1\}^n \to \{0,1\}^{n^s}\}$ be a family of functions. For a PPT adversary $\mathcal{A}$, we define*

$$\mathsf{Adv}_{F_n,W_n}(\mathcal{A}) := \left| \Pr_{\mathcal{A},w \xleftarrow{\$} W_n} [\mathcal{A}(F_n, F_n(w)) = 1] - \Pr_{u \xleftarrow{\$} \mathcal{U}_{n^s}} [\mathcal{A}(F_n, u) = 1] \right|.$$

F is called to be ε-pseudorandom generator over $\mathcal{W}$ if for any PPT adversary $\mathcal{A}$, it holds that

$$\mathsf{Adv}_{F_n,W_n}(\mathcal{A}) \leqslant \varepsilon(n).$$

If $\varepsilon = \mathsf{negl}(n)$, then we say that F is pseudorandom generator over W.

Chapter 3

ID-based Data Access Control

In this section, we propose a generalized notion of NTRU lattices called module-NTRU(MNTRU) lattices which enables to solve the dimension inflexibility of NTRU-based cryptosystems. We also show efficient generation a trapdoor over MNTRU lattices, and argue that our generalization yields better efficiency than NTRU trapdoor as well as parameter flexibility. Based on our MNTRU trapdoor, we construct a new IBE scheme as a generalization of the Gentry-Peikert-Vaikuntanathan (GPV) framework [GPV08] based on NTRU trapdoor. We also rigorously analyze the parameter choices with respect to the correctness and the security of the scheme. Our generalization derives much efficient parameter instantiation upon previous IBE scheme over MNTRU lattices.

3.1 Module-NTRU Lattices

In this section, we introduce the generalized notion of NTRU lattices described in Section 2.4.2. To give intuition, we understand the NTRU trap-

door generation by following. First, it samples short polynomials $f, g \in \mathcal{R}$, and we view this by sampling a small matrix $\mathbf{S} = \begin{bmatrix} g \\ -f \end{bmatrix}$. Then, an NTRU instance $h = g/f \in \mathcal{R}_q$ can be understood by an element obtained from a vector orthogonal to $\mathbf{S}$. In this case, such orthogonal vector is clearly $(f, g) \in \mathcal{R}^2$, and h comes from the quotient vector $(1, g/f) \in \mathcal{R}_q^2$. Finally, we extend $\mathbf{S}$ to the trapdoor $\mathbf{T}_{\mathrm{NTRU}}$ by solving the NTRU equation that satisfies $gF - fG = q$, and define $\mathbf{T}_{\mathrm{NTRU}} = \begin{pmatrix} \mathcal{A}(g) & \mathcal{A}(G) \\ -\mathcal{A}(f) & -\mathcal{A}(F) \end{pmatrix}$.

In Section 3.1.1, we elaborate the generalization of the above understanding of NTRU instance and trapdoor generation, which we call module-NTRU (MNTRU) instance and trapdoor. We will apply this new trapdoor for IBE scheme in later sections, and the Gram-Schmidt norm of the trapdoor matrix is closely related to its efficiency. Regarding this, we analyze and discuss about the Gram-Schmidt norm of the trapdoor matrix in Section 3.1.2.

3.1.1 Construction of MNTRU lattice and trapdoor

Our new construction essentially follows the above described framework for NTRU; we first set a small matrix $\mathbf{S} \in \mathcal{R}^{d \times (d-1)}$ which corresponds to $(g, -f)^t$, and consider a vector orthogonal to $\mathbf{S}$, say

$$\mathsf{det} = (\mathsf{det}_1, \cdots, \mathsf{det}_d),$$

whose name indicates, this vector is indeed computed from the determinant of submatrices of $\mathbf{S}$. Then we define a MNTRU instance by a vector $\boldsymbol{h} \in \mathcal{R}_q^{d-1}$ such that $(1, \boldsymbol{h}) = \mathsf{det}_1^{-1} \cdot \mathsf{det}$. Finally, we consider a generalized

version of NTRU equation defined by

$$\sum_{i=1}^{d} \mathsf{det}_i \cdot F_i = q,$$

and by concatenating $\boldsymbol{F} = (F_1, \cdots, F_d)$ to $\mathbf{S}$, we complete the trapdoor $\mathbf{T}_{\mathrm{MNTRU}}$ generation.

We elaborate from the generation of $\mathbf{S}$. Firstly, we sample vector of polynomials $\boldsymbol{f}_i = (f_{1,i}, \cdots, f_{d,i}) \in \mathcal{R}^d$ for $1 \leqslant i \leqslant d-1$ where each $f_{j,i}$ is a small polynomial(having small coefficients), and define a matrix $\mathbf{S} = [\boldsymbol{f}_1, \cdots, \boldsymbol{f}_{d-1}] \in \mathcal{R}^{d \times (d-1)}$, and assume that S is full-rank in R_q which happens with high probability.

To find a vector orthogonal to $\mathbf{S}$ over $\mathcal{R}$, we define $\mathbf{S}_i$ be the $(d-1) \times (d-1)$ matrix that results from deleting i-th row of $\mathbf{S}$, and define $\mathsf{det}_i = (-1)^{i-1} \cdot \det(\mathbf{S}_i)$. Then the following lemma holds.

Lemma 3.1.1. *The vector* $\mathsf{det} = (\mathsf{det}_i)_{1 \leqslant i \leqslant d}$ *satisfies* $\mathsf{det}^t \cdot \boldsymbol{S} = \mathbf{0}$ *over* $\mathcal{R}$.

Proof. We show det^t is orthogonal to each column $\boldsymbol{f}_i$ of $\mathbf{S}$ by considering a $d \times d$ matrix $\mathbf{M}_i = [\boldsymbol{f}_i \,||\, \mathbf{S}]$. Since $\mathbf{M}_i$ has the same two columns, it has determinant 0. Now the cofactor expansion by the first column implies $\det(\mathbf{M}_i) = \mathsf{det}^t \cdot \boldsymbol{f}_i$, which ends proof. □

Assuming that det_1 is invertible in $\mathcal{R}_q$ (hence $\mathbf{S}$ is full-rank in $\mathcal{R}_q$), we define the MNTRU instance $\boldsymbol{h}_{\mathrm{MNTRU}} \in \mathcal{R}_q^{d-1}$ as

$$\boldsymbol{h}_{\mathrm{MNTRU}} = (h_1, \cdots, h_{d-1}).$$

From Lemma 3.1.1, it holds that $(1, \boldsymbol{h}_{\mathrm{MNTRU}}) \cdot \mathbf{S} = \mathbf{0} \mod q$. We then define the dn-dimensional MNTRU lattice Λ_{MNTRU} associated to $\boldsymbol{h}$ and q

by

$$\Lambda_{\mathrm{MNTRU}} = \{(u_0, \cdots, u_{d-1}) \in \mathcal{R}^d : u_0 + u_1 h_1 + \cdots + u_{d-1} h_{d-1} = 0 \mod q\},$$

whose basis is given by

$$\mathbf{A}_{\mathrm{MNTRU}} := \begin{pmatrix} -\mathcal{A}(h_1) & -\mathcal{A}(h_2) & \cdots & -\mathcal{A}(h_{d-1}) & qI_n \\ I_n & O_n & \cdots & O_n & O_n \\ O_n & I_n & \cdots & O_n & O_n \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ O_n & O_n & \cdots & I_n & O_n \end{pmatrix}.$$

We proceed to the generation of the MNTRU trapdoor $\mathbf{T}_{\mathrm{MNTRU}} \in \mathbb{Z}^{dn \times dn}$ of Λ_{MNTRU}. For that, we consider the *generalized NTRU equation (MNTRU equation)* which was previously defined in [PP19], where we utilize a restricted version: Given $\mathbf{S} \in \mathcal{R}^{d \times (d-1)}$, find polynomials $F_1, ..., F_d \in \mathcal{R}$ such that

$$\sum_{i=1}^{d} \mathsf{det}_i \cdot F_i = q \tag{3.1}$$

where det_i for $1 \leqslant i \leqslant d$ are defined above. This can be done by generalizing the previous method in [HHGP+03], or applying more developed method of [PP19]. As our proof-of-concept implementation exploits the former method, we give the detailed procedure by following. Let $\mathsf{det} = (\mathsf{det}_1, \cdots, \mathsf{det}_d) \in \mathcal{R}^d$ be a vector of polynomial, and let $\phi = X^n + 1$. Our goal is to find $\boldsymbol{F} = (F_1, \cdots, F_d) \in \mathcal{R}^d$ satisfying

$$\sum_{i=1}^{d} \mathsf{det}_i \cdot F_i = q.$$

- First, compute $s_i \in \mathbb{Z}[X]$ such that

$$s_i \mathsf{det}_i = R_i \mod \phi,$$

  where $R_i \in \mathbb{Z}$ is the resultant of det_i and ϕ.

- Compute the GCD δ of R_i, with coefficients $u_i \in \mathbb{Z}$ such that

$$\sum_{i=1}^{d} u_i R_i = \delta.$$

- If δ divides q, define

$$F_i' = \frac{q \cdot u_i}{\delta} s_i.$$

The vector $\boldsymbol{F}' = (F_1', \cdots, F_d')$ may have too large size, and hence we use Babai's reduction on $\boldsymbol{F}'$ with a matrix $\mathbf{S}$, which gives much shorter solution $\boldsymbol{F} = (F_1, \cdots, F_d)$ of the MNTRU equation.

For a solution vector $\boldsymbol{F} = (F_1, ..., F_d) \in \mathcal{R}^d$ of the MNTRU equation, we set the trapdoor $\mathbf{T}_{\text{MNTRU}} \in \mathbb{Z}^{dn \times dn}$ as the concatenation of $\mathcal{A}_n(\mathbf{S})$ and $\mathcal{A}_n(\boldsymbol{F})$, i.e.,

$$\mathbf{T}_{\text{MNTRU}} := (\mathcal{A}_n(\mathbf{S}) || \mathcal{A}_n(\boldsymbol{F})).$$

We know that $(1, \boldsymbol{h}_{\text{MNTRU}}) \cdot \mathbf{S} = \mathbf{0} \mod q$ from Lemma 3.1.1, and moreover (3.1) implies that $\langle (1, \boldsymbol{h}_{\text{MNTRU}}), \boldsymbol{F} \rangle = \mathbf{0} \mod q$, and hence a lattice $\Lambda(\mathbf{T}_{\text{MNTRU}})$ is contained in Λ_{MNTRU}. Finally, Lemma 3.1.2 below says $\Lambda(\mathbf{T}_{\text{MNTRU}})$ is full-rank, which completes the construction of trapdoor $\mathbf{T}_{\text{MNTRU}}$ for the MNTRU lattice Λ_{MNTRU}.

Lemma 3.1.2. $\Lambda(\boldsymbol{T}_{MNTRU}) \supset qI_{dn}$.

Proof. We only need to show that $\Lambda_{\mathcal{R}}(\mathbf{S}||\boldsymbol{F}) \supset q\mathcal{R}^d$. Let $\boldsymbol{e}_i \in \mathcal{R}^d$ denote the

unit vector whose i-th component is 1 for $1 \leqslant i \leqslant d$. Since $\sum_{i=1}^{d} \mathsf{det}_i \cdot F_i = q$, the determinant of $\mathbf{T}_{\text{MNTRU}}$ is $(-1)^{d-1} \cdot q$. Let $M_{i,j}$ be the (i,j)-minor of $\mathbf{T}_{\text{MNTRU}}$, the determinant of $(d-1)\times(d-1)$ matrix results from deleting i-th row and j-th column of $\mathbf{T}_{\text{MNTRU}}$, and define $\boldsymbol{M}_i := (M_{i,1}, M_{i,2}, ..., M_{i,d})^t \in R^d$. Then, by the cofactor expansion, it holds that

$$\mathbf{T}_{\text{MNTRU}} \cdot \boldsymbol{M}_i = (-1)^{i-1} \cdot \det(\mathbf{T}_{\text{MNTRU}}) \cdot \boldsymbol{e}_i = \pm q \boldsymbol{e}_i,$$

which proves our claim. □

Note that Lemma 3.1.2 only implies that $\Lambda(\mathbf{T}_{\text{MNTRU}})$ is a full-rank *sublattice* of Λ_{MNTRU}, but does not guarantee that $\Lambda(\mathbf{T}_{\text{MNTRU}}) = \Lambda_{\text{MNTRU}}$, and hence $\mathbf{T}_{\text{MNTRU}}$ is not proven to be a trapdoor *basis* for Λ_{MNTRU}; recall that for NTRU case, $\mathbf{T}_{\text{NTRU}}$ is a basis of Λ_{NTRU}. We first note that it is well known(e.g., Lemma 7.1 of [MG02]) that $\mathbf{T}_{\text{MNTRU}}$ can be efficiently converted into a basis $\mathbf{B}$ of Λ_{MNTRU} such that $\|\mathbf{B}^*\| \leqslant \|\mathbf{T}^*\|$. As a more important remark, the full-rank set $\mathbf{T}_{\text{MNTRU}}$ indeed suffices for the trapdoor usage, and hence we never perform such basis-converting process in our IBE scheme.

Hardness Assumption

The original NTRU trapdoor obtains its hardness from NTRU assumption that as, for two small random polynomials f and g in $\mathcal{R}$, their quotient $h = fg^{-1} \in \mathcal{R}_q$ is indistinguishable from uniform element in $\mathcal{R}_q$. For our case, we can establish a similar MNTRU assumption, saying

$$\boldsymbol{h}_{\text{MNTRU}} = \mathsf{det}_1^{-1} \cdot (\mathsf{det}_2, \cdots, \mathsf{det}_d) \in \mathcal{R}_q^{d-1}$$

is indistinguishable from a uniform vector in $\mathcal{R}_q^{d-1}$.

In fact, what we exactly need is somewhat weaker notion; to apply the GPV framework, we require SIS is hard over a random choice of $\mathbf{A}$. Thus, our following IBE scheme is secure under somewhat mild assumption that SIS is hard over $\mathbf{A}_{\text{MNTRU}}$ on average, where the randomness is from the random choice of $\mathbf{S}$.

3.1.2 Minimize the Gram-Schmidt norm

For an IBE scheme in GPV framework, the users' secret key issue involves a discrete Gaussian sampling over $\Lambda(\mathbf{T}_{\text{MNTRU}})$. As known discrete Gaussian samplers sample Gaussian having size proportional to $\|\mathbf{T}^*_{\text{MNTRU}}\|$, it is quite important to set $\mathbf{T}_{\text{MNTRU}}$ to have small Gram-Schmidt norm $\|\mathbf{T}^*_{\text{MNTRU}}\|$. In this regard, we now explain how we choose $\mathbf{S} \in \mathcal{R}$ to minimize $\|\mathbf{T}^*_{\text{MNTRU}}\|$.

We start from the following lemma adapted from Lemma 2 of [DLP14] that says for MNTRU trapdoor, we only need to see d Gram-Schmidt norms to determine $\|\mathbf{T}^*_{\text{MNTRU}}\|$.

Lemma 3.1.3. *Let $\boldsymbol{T}_{MNTRU} = [\boldsymbol{t}_1 \cdots \boldsymbol{t}_{dn}]$ be the MNTRU trapdoor. Then*

$$\|\boldsymbol{T}^*_{MNTRU}\| = \max\{\|\boldsymbol{t}^*_1\|, \|\boldsymbol{t}^*_{n+1}\|, \cdots \|\boldsymbol{t}^*_{(d-1)n+1}\|\}$$

Intuitively, we expect that the minimal occurs when

$$\|\boldsymbol{t}^*_1\| = \|\boldsymbol{t}^*_{n+1}\| = \cdots = \|\boldsymbol{t}^*_{(d-2)n+1}\| = \|\boldsymbol{t}^*_{(d-1)n+1}\|.$$

Since the first $d-1$ norms depend on our choice of $\boldsymbol{f}_i \in \mathcal{R}^d$, we first choose

$\boldsymbol{f}_{i+1} \in \mathcal{R}^d$ for $1 \leqslant i \leqslant d-2$ for the first $d-2$ equality by

$$\|\boldsymbol{t}_{in+1}\| = \sqrt{\frac{d}{d-i}} \cdot \|\boldsymbol{t}_1\|. \tag{3.2}$$

As underlying idea for this choice, we see that $\boldsymbol{t}^*_{in+1}$ is a projection of $\boldsymbol{t}_{in+1}$ (of dimension dn) over a subspace of dimension $(d-i)n$, and hence random choice of $\boldsymbol{f}_i$ implies

$$\|\boldsymbol{t}^*_{(i-1)n+1}\| = \sqrt{\frac{d-i+1}{d}} \cdot \|\boldsymbol{t}_{(i-1)n+1}\|.$$

We experimentally check this choice of $\boldsymbol{f}_i$ indeed implies

$$\|\boldsymbol{t}^*_1\| = \|\boldsymbol{t}^*_{n+1}\| = \cdots = \|\boldsymbol{t}^*_{(d-2)n+1}\|,$$

and Figure 3.1 shows the result with $d = 4$ case.

Finally the last one $\|\boldsymbol{t}^*_{(d-1)n+1}\|$ depends on our choice of $\mathbf{S} = [\boldsymbol{f}_1, \cdots, \boldsymbol{f}_{d-1}]$, and we investigate the optimal choice of $\|\boldsymbol{t}_1\|$ while varying $\|\boldsymbol{t}_1\|$. We presume that such optimal choice is represented by $c_d \cdot q^{1/d}$ for some constant c_d that depends only on d, which implies the Gram-Schmidt norm of $\mathbf{T}_{\mathrm{MNTRU}}$ can be reached to

$$\|\mathbf{T}^*_{\mathrm{MNTRU}}\| \leqslant c_d \cdot q^{1/d}.$$

Note that this is consistent with the known result of [DLP14] with $c_2 = \sqrt{e/2} \approx 1.1658$, which is also provided with heuristic analysis. Regarding this, we experimentally verify that it holds for $c_3 \approx 1.2$ as Figure 3.2.

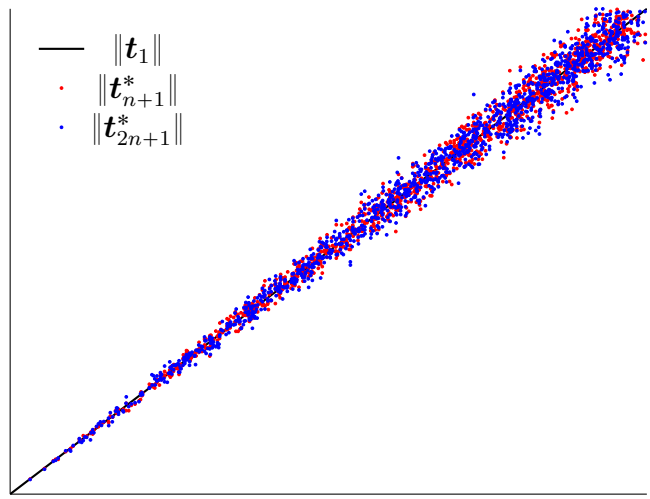

Figure 3.1: $\|\boldsymbol{t}^*_{in+1}\|$ values with $\|\boldsymbol{t}_{in+1}\| = \sqrt{\frac{d}{d-i}} \cdot \|\boldsymbol{t}_1\|$ for $i = 1, 2$, with $(d, n, q) = (4, 256, 2^{27})$

3.2 IBE-Scheme from Module-NTRU

In this section, we describe our IBE scheme, whose security is based on MNTRU and Module-LWE.

3.2.1 Scheme Construction

We start from master key generation procedure KeyGen. It basically generates the MNTRU instance $\boldsymbol{h} = (h_1, \cdots, h_{d-1}) \in \mathcal{R}_q^{d-1}$ as the master public key and the MNTRU trapdoor matrix $\mathbf{T}_{\text{MNTRU}} \in \mathbb{Z}^{dn \times dn}$ as the master secret key. The master secret key elements are sampled according to Section 3.1.2, which implies

$$\|\mathbf{T}^*_{\text{MNTRU}}\| = c_d \cdot q^{1/d}.$$

The detailed procedure is given by Algorithm 1.

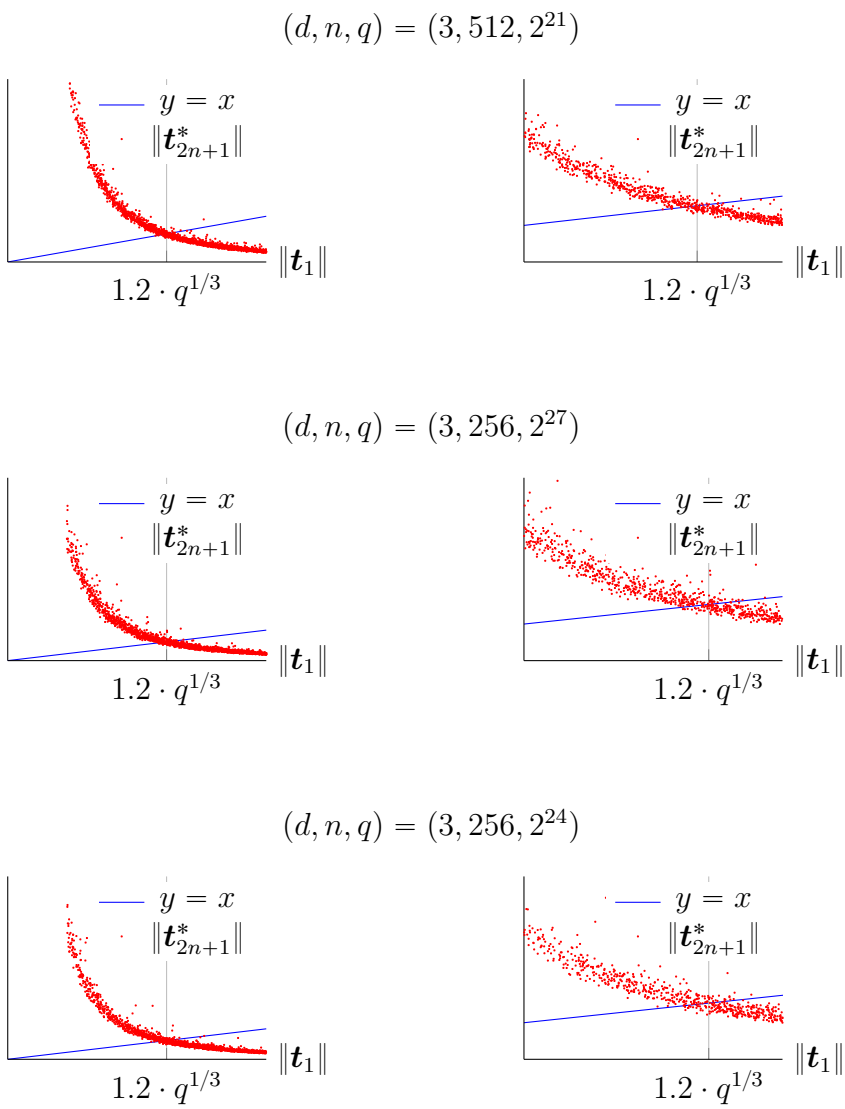

Figure 3.2: Values of $\|\boldsymbol{t}^*_{(d-1)n+1}\|$ for $d = 3$, which indicates $c_3 \approx 1.2$ regardless of n and q.

Algorithm 1: KeyGen

Input : n, q, d

Output: $\mathsf{MPK} = \boldsymbol{h} \in R_q^{d-1}$ and $\mathsf{MSK} = \mathbf{T}_{\mathrm{MNTRU}} \in \mathbb{Z}^{dn \times dn}$

for $i = 1$ *to* $d-1$ **do**
  $\sigma_i \leftarrow \sqrt{\frac{d}{d-i+1}} c_d \cdot q^{1/d}/\sqrt{dN}$;
  $\boldsymbol{y} = (\boldsymbol{y}_1 || \boldsymbol{y}_2) \leftarrow BKZ_{\delta_0}(\Lambda_{q,c}^{\perp}(A_1))$;
  $\boldsymbol{f}_i \leftarrow (f_{1,i}, \cdots, f_{d,i})$ where each coefficient of $f_{j,i} \in \mathcal{R}$ is sampled from $\mathcal{D}_{\mathbb{Z},\sigma_i}$
end
$\mathbf{S} \leftarrow [\boldsymbol{f}_1, \cdots, \boldsymbol{f}_{d-1}]$;
$\mathsf{det} \leftarrow (\mathsf{det}_1, \cdots, \mathsf{det}_d)$ where $\mathsf{det}_i = (-1)^{i-1} \cdot \det(\mathbf{S}_i)$;
$\boldsymbol{h} \leftarrow \mathsf{det}^{-1} \cdot (\mathsf{det}_2, \cdots, \mathsf{det}_d) \in \mathcal{R}_q^{d-1}$;
Find a solution $\boldsymbol{F} = (F_1, \cdots, F_d) \in \mathcal{R}^d$ of the MNTRU equation $\sum_{i=1}^{d} \mathsf{det}_i \cdot F_i = q$;
$\mathbf{T} \leftarrow [\mathcal{A}(\mathbf{S}) || \mathcal{A}(\boldsymbol{F})]$;
return $\mathsf{MPK} = \boldsymbol{h}$ and $\mathsf{MSK} = \mathbf{T}$

The extract procedure issues the user secret key sk_{id} valid for user *id*. The main task for this is sampling short $\boldsymbol{s} \in \mathcal{R}^d$ such that

$$\langle \boldsymbol{s}, (1, \boldsymbol{h}) \rangle = H(id) \mod q$$

where $H : \{0,1\}^* \to \mathcal{R}_q$ is some hash function modeled as a random oracle. This vector $\boldsymbol{s}$ is computed by Gaussian sampling over Λ_{MNTRU}, and we use $\mathsf{GaussianSampler}$ with the master secret key $\mathbf{T}_{\mathrm{MNTRU}}$. The standard deviation σ is chosen to yield KL Divergence of $\mathsf{GaussianSampler}(\mathbf{T}_{\mathrm{MNTRU}}, \sigma)$ and the ideal discrete Gaussian $\mathcal{D}_{\Lambda(\mathbf{T}_{\mathrm{MNTRU}}),\sigma}$ less than $2^{-\lambda}$. It is given by

$\sigma = \eta'_\varepsilon(\mathbb{Z}) \cdot \|\mathbf{T}^*_{\mathrm{MNTRU}}\|$ where $\varepsilon = 2^{-\lambda/2}/(2\sqrt{2} \cdot dn)$, and more precisely

$$\sigma \approx \frac{c_d}{\pi} \cdot \sqrt{\frac{\ln 2}{2}\left(\frac{\lambda}{2} + \log_2(4\sqrt{2} \cdot dn)\right)} \cdot q^{1/d}. \tag{3.3}$$

We also remark that this extract procedure should be *stateful*, *i.e.,* it should stores every previously issued user secret keys, otherwise our scheme becomes insecure by repeated queries on the same *id*; actually, every IBE scheme based on GPV framework share the same feature, and some *stateless* variants are already argued in previous works. For simplicity we omit them and refer [GPV08]. The detailed procedure can be found in Algorithm 2 below.

Algorithm 2: Extract

Input : An identity *id*, the master secret key $\mathbf{T}$, the master public key $\boldsymbol{h}$ and a hash function $H : \{0,1\}^* \to \mathcal{R}_q$
Output: A user secret key $\mathsf{sk}_{id} \in \mathcal{R}^{d-1}$

if *id is previously queried* **then**
  return sk_{id} in local storage
end
else
  $\boldsymbol{t} \leftarrow (H(id), 0, \cdots, 0) \in R_q^d$;
  $\sigma \leftarrow \frac{c_d}{\pi} \cdot \sqrt{\frac{\ln 2}{2}\left(\frac{\lambda}{2} + \log_2(4\sqrt{2} \cdot dn)\right)} \cdot q^{1/d}$;
  $\boldsymbol{c} \leftarrow \mathsf{GaussisanSampler}(\mathbf{T}, \sigma, \boldsymbol{t})$;
  $\boldsymbol{s} = (s_0, s_1, \cdots, s_{d-1}) \leftarrow \boldsymbol{t} - \boldsymbol{c}$;
  Add $\mathsf{sk}_{id} = (s_1, \cdots, s_{d-1})$ in local storage;
  return sk_{id}
end

Our encryption and decryption are done in the same manner to Module-LWE based encryption. In particular, polynomials r, e_i are uniformly sam-

pled from $\{-1, 0, 1\}^n$. Moreover, our IBE scheme also combines KEM and one-time-pad (OTP) as in [DLP14]. This combination of OTP is necessary for our case where the width parameter σ is chosen to have negligible KL divergence of Gaussian sampler; KL divergence argument only applies for search problems, and without the use of OTP, we cannot guarantee indistinguishability based security of our scheme.

Algorithm 3: Encrypt

Input : An identity id, a message $\mu \in \{0,1\}^m$, the master public key $\boldsymbol{h} \in R_q^{d-1}$, hash functions $H : \{0,1\}^* \to R_q$ and $H' : \{0,1\}^n \to \{0,1\}^m$

Output: A ciphertext $C = (\boldsymbol{c}, c')$ where $\boldsymbol{c} \in R_q^d$ and $c' \in \{0,1\}^m$.

$r, e_i \leftarrow \{-1, 0, 1\}^n$ for $0 \leqslant i \leqslant d-1$; $k \leftarrow \{0,1\}^n$;
$t \leftarrow H(id)$;
$c_0 \leftarrow rt + e_0 + \left\lfloor \frac{q}{2} \right\rceil \cdot k$;
$c_0 \leftarrow 2^{\lceil \log_2 q \rceil - 3} \cdot \left\lfloor \frac{c_0}{2^{\lceil \log_2 q \rceil - 3}} \right\rceil$;
$\boldsymbol{c} \leftarrow (c_0, c_1, \cdots, c_{d-1})$ where $c_i = rh_i + e_i$ for $1 \leqslant i \leqslant d-1$;
$c' \leftarrow \mu \oplus H'(k)$;
return $C = (\boldsymbol{c}, c')$

Algorithm 4: Decrypt

Input : A ciphertext $C = (\boldsymbol{c}, c')$, a user secret key $\mathsf{sk}_{id} \in \mathcal{R}^{d-1}$, and hash functions $H : \{0,1\}^* \to \mathcal{R}_q$ and $H' : \{0,1\}^n \to \{0,1\}^m$

Output: A message $\mu \in \{0,1\}^m$

$\boldsymbol{s}' = (1, -\mathsf{sk}_{id})$;
$w \leftarrow \langle \boldsymbol{c}, \boldsymbol{s}' \rangle$;
$k \leftarrow \left\lfloor \frac{2}{q} \cdot w \right\rceil$;
return $m \leftarrow c \oplus H'(k)$

For the decryption correctness, observe that

$$w = \langle \boldsymbol{c}, (1, -\mathsf{sk}_{id}) \rangle = \left\lfloor \frac{q}{2} \right\rceil \cdot m + e_0 + rs_0 - \sum_{i=1}^{d-1} e_i s_i.$$

Then each coefficient of the error polynomial $e_0 + rs_0 - \sum_{i=1}^{d-1} e_i s_i$ should lie over $(-q/4, q/4)$. We first estimate the coefficient size of the error polynomial by approximating it into (continuous) Gaussian distribution having the same variance. Precisely, it is assumed to behave like 0-centered Gaussian with variance $\frac{2}{3}(\|\mathsf{sk}_{id}\|^2 + 1)$. Using a tail bound for Gaussian distribution, we have the following condition for correctness:

$$q \geqslant \frac{32\sqrt{\lambda \ln 2}}{3\sqrt{3}} \cdot \|\mathsf{sk}_{id}\|. \tag{3.4}$$

Moreover, as in [DLP14], one can reduce the size of ciphertext by sending only a few highest order bits of c_0, which not much harm the correctness of decryption.

3.2.2 Security Analysis by Attack Algorithms

In this section, we give security analysis of our IBE scheme based on the following facts from the literature. First, adapted from [PAFZ19]'s argument, if an N-dimensional lattice Λ is known to have an unusually short vector $\boldsymbol{v}$ whose size is evidently smaller than Gaussian Heuristic $\left(\sqrt{\frac{N}{2\pi e}} \cdot \det(\Lambda)^{1/N}\right)$, it can be found by BKZ with blocksize β satisfying

$$0.75\sqrt{\beta/N} \cdot \|\boldsymbol{v}\| \leqslant \delta_0^{2\beta - N} \det(\Lambda)^{1/N} \tag{3.5}$$

where the root Hermite factor δ_0 of BKZ_β is given by $\left(\frac{\beta}{2\pi e}(\pi\beta)^{\frac{1}{\beta}}\right)^{\frac{1}{2(\beta-1)}}$ [Che13].

On the other hand, for any N-dimensional lattice Λ, if one wants to find a vector $\boldsymbol{v}$ whose size is larger than $\det(\Lambda)^{1/N}$, the required root Hermite factor δ_0 is determined by

$$\delta_0^N \leqslant \frac{\|\boldsymbol{v}\|}{\det(\Lambda)^{1/N}}. \tag{3.6}$$

Based on these facts, we mount lattice attacks on several possible attack points.

Master Key Recovery

One may try to recover MSK from MPK, by finding an unusually short vector $\boldsymbol{f}_i$ in a lattice with a basis $\mathbf{A}_{\mathrm{MNTRU}}$. Since the short vector $\boldsymbol{f}_i$ is chosen to have norm smaller than $\sqrt{\frac{d}{2}} \cdot c_d \cdot q^{1/d}$, (3.5) implies that

$$\frac{0.75\sqrt{\frac{\beta}{dn}} \cdot \sqrt{\frac{d}{2}} \cdot c_d \cdot q^{1/d}}{q^{1/d}} \approx 0.75 \cdot c_d \sqrt{\frac{\beta}{2n}} = \delta_0^{2\beta - dn}.$$

User Key Recovery

The attacker can try to obtain an user secret key id from $\mathsf{MPK} = \boldsymbol{h}$, which involves finding any short $\boldsymbol{s} \in \mathcal{R}^d$ satisfying $\langle \boldsymbol{s}, (1, \boldsymbol{a}) \rangle = H(id)$. This can be done by finding a short vector $(\boldsymbol{s}, 1)$ in a $dn + 1$-dimensional lattice with determinant q^n. For correct decryption, the target vector norm would be approximately $\sqrt{dn} \cdot \sigma$ where σ comes from (3.3). Then (3.6) gives a condition

$$\frac{\sqrt{dn} \cdot \sigma}{q^{n/(dn+1)}} \approx \frac{\sqrt{dn} \cdot \sigma}{q^{1/d}} = \delta_0^{dn}.$$

IND-CPA security

Our ciphertext is of the form

$$(c_0, c_1, \cdots, c_{d-1}) = (rt + e_0, rh_1 + e_1, \cdots rh_{d-1} + e_{d-1})$$

for $\mathsf{MPK} = \boldsymbol{h}$. Like the above user key recovery case, one can try to find the $dn+1$-dimensional vector $(e_0, \cdots, e_{d-1}, 1)$ in a lattice with determinant q^n. Since we know the unusual short vector $(e_0, \cdots, e_{d-1}, 1)$ of size $\approx \sqrt{2dn/3}$ in the lattice, we apply (3.5)

$$0.75\sqrt{\frac{\beta}{dn}} \cdot \sqrt{2dn/3} = 0.75\sqrt{\frac{2\beta}{3}} \leqslant \delta_0^{2\beta - dn} q^{1/d}.$$

3.2.3 Parameter Selections

We now set a concrete parameter (d, n, q), and compare our scheme with previous results. First of all, we note that it should be noted that if one wants to use MNTRU dimension d, the master key generation involves a sampling from a discrete Gaussian with width $\sigma \approx q^{1/d}/\sqrt{dn}$. However for $d > 3$ case, σ becomes extremely small (less than 0.5) for our interest modulus q and dimension n ranges. Thus, in order to hedge against any possible problems regarding this extremely small discrete Gaussian, we conservatively consider only small d, explicitly $d = 3$. Moreover besides this discrete Gaussian sampling issue, too large d implies too small width parameter σ, and the resulting secret matrix $\mathbf{S}$ would be almost zero matrix, which can be find out by simple exhaustive search.

One may use some portions of vectors among $u_1, \cdots, u_{d-1}$ and v, but we also have the same result.

In this regard, we instantiate our scheme with $d = 3$, with modulus parameter $q = 2^{19}$ for $n = 512$, which satisfies the correctness condition (3.4). Upon our security analysis of Section 3.2.2, the minimal block size for attacking our scheme is 506; Master key recovery requires $\beta \geqslant 714$, and user key recovery requires $\beta \geqslant 612$, and IND-CPA security requires $\beta \geqslant 506$. According to methodology of [ADPS16], we estimate BKZ call with block size β costs $2^{0.292\beta}$ time, and hence our instantiation provides about 147 security level. For a pair comparison, we re-evaluate security of [DLP14] parameter ($d = 2, n = 1024, q = 2^{27}$) according to our renewed security analysis of Section 3.2.2; Master key recovery requires $\beta \geqslant 908$, and user key recovery requires $\beta \geqslant 867$, and IND-CPA security requires $\beta \geqslant 300$.

Finally we also compare key sizes and ciphertext size. Clearly the ciphertext and master public key consists of $d-1$ elements in R_q, so their bitsizes are $(d-1)n(\lfloor \log_2 q \rfloor + 1)$. Next, the user secret key consists of $d-1$ elements in $\mathcal{R}$ whose coefficients are sampled from a discrete Gaussian of standard deviation $\sigma = \frac{c_d}{\pi} \cdot \sqrt{\frac{\ln 2}{2}\left(\frac{\lambda}{2} + \log_2(4\sqrt{2} \cdot dn)\right)} \cdot q^{1/d}$ from (3.3); for our case $\sigma \approx 2.33 \cdot q^{1/3}$, and [DLP14] case $\sigma \approx 2.28 \cdot \sqrt{q}$ (with $\lambda = 192$). This can be stored in various ways, and we follow FALCON's method that requires about $(d-1)n \cdot (\lfloor \log_2(\sigma) \rfloor + 2)$.

We also check our proposal by a proof-of-concept implementation, and experimental results consisting speed results and concrete bit-sizes can be found in Table 3.1 below. However, we remark again that this implementation is literally for proof-of-concept, and our superiority on speed results over [DLP14] should not be taken seriously.

*In [DLP14], this parameter set was claimed to have 192-bit security based on their own security analysis. However we adapt the latest, rather conservative security analysis of literature, and it concludes 87-bit security for that parameter set.

	[DLP14]	**Ours**
$(d, n, \log_2 q)$	(2, 1024, 26)	(3, 512, 19)
Bit-security	87*	147
Ciphertext size (bytes)	3328	2432
Master pk size (bytes)	3328	2432
User sk size (bytes)	2048	1152
User KeyGen (ms)	22.02	12.6
Enc + Dec (ms)	4.9	1.6

Table 3.1: Comparison between [DLP14] and our scheme. Both experiments are done on Intel (R) Xeon (R) Silver 4144 processor (2.20GHz CPU). Full implementation can be found on `github.com/Yongyongha/Module-NTRU`.

3.3 Application to Signature

Our MNTRU trapdoor can be used for building a signature scheme. Let n be a power-of-two integer, $d \geqslant 2$ be a MNTRU dimension and $q > 0$ be modulus. In this case, we use the same keygen algorithm to output a public verification key $\mathsf{VK} = \boldsymbol{a}$ and a secret signing key $\mathsf{SK} = \mathbf{T}_{\text{MNTRU}}$. For a message μ, the signing procedure runs the $\mathsf{extract}$ algorithm with $t = H(\mu)$ to output a sign $\boldsymbol{s}$. The corresponding verification procedure checks whether $\boldsymbol{s}$ is short and $\langle \boldsymbol{s}, (1, \boldsymbol{a}) \rangle = H(\mu)$. The public key size would be $(d-1)n \cdot \lceil \log_2 q \rceil$, and the signature size would be $(d-1)n \cdot (\lfloor \log_2(\sigma) \rfloor + 2)$. FALCON chooses $\sigma \approx 1.312 \cdot \|\mathbf{T}^*\|$ from Rényi divergence argument due to [Pre17], which translates into $\sigma \approx 1.55 \cdot \sqrt{q}$ in FALCON case, and $\sigma \approx 1.58 \cdot q^{1/3}$ in our case.

For the signature usage, there is no encryption phase and we only consider the secret key recovery (the master key recovery in IBE) and the signature forgery (the user key recovery in IBE) attacks. In this case, one

can check that the other attacks are only relevant to the total dimension $N = nd$, in other words, q is irrelevant to security level. Thus, under the same security level, the ring dimension n is proportional to $1/d$ and hence we conclude that the pk size is asymptotically proportional to $1 - \frac{1}{d}$, and the sign size is asymptotically proportional to $\frac{d-1}{d^2}$. However, regarding the concrete parameters, such asymptotic decreases in sig size is not so huge due to the small choice of q, and indeed the expected size of signature becomes rather larger than the NTRU case due to the constant terms. For example, FALCON chooses q to be the smallest prime such that $q = 1 \mod 2n$ (12289) for $n = 512$ and 1024 case, and $q = 1 \mod 3n$ (18433) for $n = 768$ case[†]. We focus on $n = 768$ and $d = 2$ case having total dimension 1536, where we can divide the same total dimension by $n = 512$ and $d = 3$, and use modulus $q = 12289$. Note that this two parameter sets provide the same security levels, as they have the same total dimension. The concrete sizes are compared in Table 3.2.

	[PAFZ19]	**Ours**
(d, n, q)	(2, 768, 18433)	(3, 512, 12289)
Bit-security	195	195
VK size (bytes)	1440	1792
Sig size (bytes)	864	892

Table 3.2: Comparison between [PAFZ19] and our scheme

However, we remark that our generalization can still contributes for digital signatures by introducing parameter flexibility with power-of-two dimensional rings. We leave an open question that whether many optimization techniques for power-of-two ring case are applicable, which may lead to practical (M)NTRU-based cryptosystem like MLWE-based

schemes [BDK+18, DKL+18] in Post-Quantum Cryptography realm.

[†]This is for the purpose of using *number theoretic transform*(NTT), which enables fast operations on R_q.

Chapter 4

Noisy Key Cryptosystem

We propose reusable fuzzy extractors for Hamming distance on binary alphabet, under the adversary model that the perturbation of each multiple reading for reusability is controlled by adversary as in [Boy04, WLG19]. Our core ingredient is a special function family so-called *local functions* which have been mainly considered for simple constructions of fundamental cryptographic primitives; one-way functions and pseudorandom generators. Indeed, we obtain the reusable security of our fuzzy extractors from the one-wayness and pseudorandomness of the local functions.

We then propose two different approaches for achieving the functionality of fuzzy extractor. The first one exploits the fact that local functions approximately preserves the distance of inputs due to its simple structure, and the second one uses a rather complex argument that says the knowledge of an approximate value of the preimage of local function efficiently leads to recovery of the exact preimage value. Based on these ideas, we construct fuzzy extractors where each scheme can be shown over different type of random sources, where all of them can tolerate linear fraction of

error if instantiated with a proper local function, assuming that they enjoy the sufficient security level.

Due to the limited understanding for hardness of local functions over non-uniform source in literature, we cannot specify a concrete random source other than uniform source where we have local one-way functions (OWF) or local pseudorandom generators (PRG). In this regard, we also present arguments about the cryptographic hardness of local functions over non-uniform source. From this argument we derive plausibility of local OWF and weaker variant of local PRG over some non-uniform sources, which reinforce the security ground of our fuzzy extractor schemes.

4.1 Reusable Fuzzy Extractors

A (n, κ, t, δ)-fuzzy extractor is an algorithm tuple (Init (*initialize*), Gen (*generate*), Rep (*reproduce*)) satisfying

- Init takes an input security parameter 1^λ, and outputs public parameter pp.
- Gen takes public parameter pp and a string $w \in \{0,1\}^n$, and outputs an extracted string $\boldsymbol{r} \in \{0,1\}^\kappa$ and helper $\mathsf{H} \in \{0,1\}^*$.
- Rep takes public parameter pp and a string $w' \in \{0,1\}^n$ and H, and outputs a string $\boldsymbol{r'} \in \{0,1\}^\kappa$ or $\perp$.
- For the correctness, for $\boldsymbol{w}, \boldsymbol{w'} \in \{0,1\}^n$ such that $\mathsf{HD}(\boldsymbol{w}, \boldsymbol{w'}) \leqslant t \cdot n^*$and $\mathsf{pp} \leftarrow \mathsf{Init}(1^\lambda)$ and $(\boldsymbol{r}, \mathsf{H}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})$, it holds that

$$\Pr[\mathsf{Rep}(\mathsf{pp}, \boldsymbol{w}, \mathsf{H}) = \boldsymbol{r}] \geqslant 1 - \delta$$

where the randomness is over the choice of $\boldsymbol{w}'$, and algorithms Gen and Rep.

Remark. The original definition requires *worst-case* correctness on $\boldsymbol{w}'$, that is, the correctness requires to hold for any $\boldsymbol{w}'$ such that $\mathsf{HD}(\boldsymbol{w}, \boldsymbol{w}') \leqslant t \cdot n$. We slightly weaken this condition to *average-case*, and note that this is already implicitly considered in [CFP+16] for their correctness analysis.

Reusable security

For the security notion for reusability, we consider two definitions based on indistinguishability (IND) and one-wayness (OW), where the first one is considered much often in literature.

Let $\mathcal{W}$ be a family of probability distributions over $\{0,1\}^{n\dagger}$. For an adversary $\mathcal{A}$, it plays $\mathsf{Exp}^{W,\rho}_{\mathsf{IND-reu}}$ (resp, $\mathsf{Exp}^{W,\rho}_{\mathsf{OW-reu}}$) by querying Init followed by at most ρ times of Chal and returns β'(resp, $\boldsymbol{r}'$), an input of Fin. See Figure 4.1 below for the definition of each procedure.

We say that a fuzzy extractor is (ρ, ε)-IND-reusable over $\mathcal{W}$ if for any distribution $W \in \mathcal{W}$, and for any PPT adversary $\mathcal{A}$

$$\left| \Pr[\mathsf{Exp}^{W,\rho}_{\mathsf{IND-reu}}(\mathcal{A}) = 1] - \frac{1}{2} \right| \leqslant \varepsilon(\lambda).$$

Similarly a fuzzy extractor is called (ρ, ε)-OW-reusable over $\mathcal{W}$ if for any distribution $W \in \mathcal{W}$, and for any PPT adversary $\mathcal{A}$

$$\Pr[\mathsf{Exp}^{W,\rho}_{\mathsf{OW-reu}}(\mathcal{A}) = 1] \leqslant \varepsilon(\lambda).$$

*In other literature, t usually denotes the error value itself. Note that in our definition t denotes the *ratio* of error, and its meaningful choice is clearly $t \in [0, 0.5)$.

If the advantage ε is $\mathsf{negl}(\lambda)$, we simply say a fuzzy extractor ρ-reusable for both cases.

$\underline{\mathsf{Init}(\mathsf{pp})}$:

1. $\boldsymbol{w} \leftarrow W$
2. $(\boldsymbol{b}, \mathsf{H}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})$
3. $\beta \leftarrow \{0,1\}$
4. If $\beta = 1$, Return $(\boldsymbol{b}, \mathsf{H})$
5. Else, $\boldsymbol{u} \leftarrow \{0,1\}^{\kappa}$, Return $(\boldsymbol{u}, \mathsf{H})$

$\underline{\mathsf{Chal}(\boldsymbol{\delta}_k)}$:

1. If $\mathsf{HW}(\boldsymbol{\delta}_k) > t \cdot n$, Return $\perp$
2. $(\boldsymbol{b}_k, \mathsf{H}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \boldsymbol{w} + \boldsymbol{\delta}_k)$
3. Return (b_k, H_k)

$\underline{\mathsf{Fin}(\beta')}$:

1. If $\beta = \beta'$, Return 1
2. Else, Return 0

$\underline{\mathsf{Init}(\mathsf{pp})}$:

1. $\boldsymbol{w} \leftarrow W$
2. $(\boldsymbol{b}, \mathsf{H}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})$
3. Return H

$\underline{\mathsf{Chal}(\boldsymbol{\delta}_k)}$:

1. If $\mathsf{HW}(\boldsymbol{\delta}_k) > t \cdot n$, Return $\perp$
2. $(\boldsymbol{b}_k, \mathsf{H}) \leftarrow \mathsf{Gen}(\mathsf{pp}, \boldsymbol{w} + \boldsymbol{\delta}_k)$
3. Return (b_k, H_k)

$\underline{\mathsf{Fin}(\boldsymbol{b}' \in \{0,1\}^{\kappa})}$:

1. If $\boldsymbol{b} = \boldsymbol{b}'$, Return 1
2. Else, Return 0

Figure 4.1: Left: $\mathsf{Exp}^{W,\rho}_{\mathsf{IND-reu}}$, Right: $\mathsf{Exp}^{W,\rho}_{\mathsf{OW-reu}}$

[†]Fuzzy extractor can be generally defined over any metric space, for instance set difference metric, but we only focus on binary string with Hamming distance case.

4.2 Local Functions

A random local function is determined by a boolean function $P : \{0,1\}^\ell \to \{0,1\}$ called *predicate*. Let n be input length and m output length. We sample $I_j = \{i_{j,1}, \cdots, i_{j,\ell}\}$ by random ℓ-element subsets of $[n]$ for $1 \leqslant j \leqslant m$, and define an index set $\mathcal{I} = \{I_j : 1 \leqslant j \leqslant m\}$. For $\boldsymbol{x} = (x_i)_{i=1}^n \in \{0,1\}^n$, the local function $\mathsf{LF}_{P,\mathcal{I}} : \{0,1\}^n \to \{0,1\}^m$ is computed as follows:

$$\mathsf{LF}_{P,\mathcal{I}}^m(\boldsymbol{x}) := \left(P(x_{i_{j,1}}, \cdots, x_{i_{j,\ell}})\right)_{1 \leqslant j \leqslant m},$$

along with the indices sets and predicate as public parameters. In this construction ℓ is called the *locality*.

4.2.1 Hardness over Non-uniform Sources

The local functions have been thoroughly researched in the cryptographic literature after argued by Goldreich [Gol00] to use them as simple one-way functions. Still, most of study focus on the hardness when the inputs are chosen uniform randomly, and their hardness has never been discussed over non-uniform source to the author's best knowledge. Here we discuss and give some clues for their security over non-uniform sources. We refer the beautiful survey [App16] by Applebaum to readers for more details study for cryptographic hardness of local random functions over uniform source.

Pseudorandomness of Local Functions over Non-uniform Source

We give some evidences that random local functions does not likely to be a strong PRGs, but it is reasonable to assume that any adversary may have a bounded advantage.

Let W be a source that is possibly non-uniform. First of all, consider the distribution $\{\mathsf{LF}^{(k)}(\boldsymbol{w}) : \boldsymbol{w} \leftarrow W\}$ of k-th coordinate. This distribution may not be balanced (i.e. the probability that occurs 0 and 1 are different). Further, to be secure against $\mathbb{F}_2$-linear attacks the distribution $\{\mathsf{LF}(\boldsymbol{w}) : \boldsymbol{w} \leftarrow W\}$ should be a negligible-biased distribution, which is not true in general. Thus the PRG assumption on LF over W seems to be false.

However, in our fuzzy extractor construction, the adversary will be only given a single sample $\mathsf{LF}(\boldsymbol{w})$, whereas most of distinguishing attacks inherently use the standard hybrid argument to inspect statistical properties of distinguishing targets; for example it is biased or not. Usually the hybrid argument makes a lose of advantage by a factor N to amplify the number of samples to N, and thus it may reasonable to assume that random (flipping) local functions is, say, 1/3-PRG.

One-wayness of Local Functions over Non-uniform Source.

We discuss here that the one-wayness of local functions over non-uniform source seems to be hard, even we weaken the goal of adversary to get an approximate inversion. This supports our weak pseudorandomness assumption as well. The hardness of inversion, or even *approximate* inversion of local function is the minimal requirement for security of our fuzzy extractor, since the input of local functions in our construction would be one's secret noise source, say human's biometrics. Fortunately, both problems seem to be hard even for non-uniform sources. We give some evidences for them.

The one-wayness of local functions over non-uniform source is supported by the self-reducibility shown by Bogdanov and Rosen [BR13, Theorem 6.1]. The self-reducibility of local functions states that, roughly, if the

inversion problem of $\mathsf{LF}(\boldsymbol{x})$ is hard in a certain level (i.e. sub-exponential to n) for a small fraction of seed $\boldsymbol{x}$, then for all but a small fraction of seeds the inversion problem of $\mathsf{LF}(\boldsymbol{x})$ also enjoy the similar level of hardness as well. Therefore the one-wayness of local functions enjoys all-or-nothing flavor: Either all but small inputs are hard to invert, or all but small inputs are easy to invert. Thus by assuming the one-wayness of local functions over uniform source, the one-wayness of local functions is likely to hold even for non-uniform source as well, with respect to the certain level of hardness and choice of underlying indices.

Further, Bogdanov and Qiao [BQ12] showed that the hardness of inversion implies the hardness of *approximate* inversion, or more formally the following theorem.

Theorem 4.2.1 ([BQ12, Theorem 1.3])**.** *Let m, n, ℓ be integers and $\mu > 0$, and $\mathsf{LF} : \{0,1\}^n \to \{0,1\}^m$ a randomly chosen local function with locality ℓ. If $m \geqslant n \cdot (k/\mu)^{2\ell}$ for a universal constant k, then there is an efficient algorithm given $\mathsf{LF}(\boldsymbol{x})$ and $\boldsymbol{x}'$ such that $\mathsf{HD}(\boldsymbol{x}, \boldsymbol{x}') \leqslant (1/2 - \mu)n$ that recovers $\boldsymbol{x}$ with probability $1 - o(1)$, where the probability is over the choice of the random local function and $\boldsymbol{x}$.*

We denote the algorithm in this theorem by BQ. Note that the probability can be improved to $1 - O(n^{-r})$ for $r \leqslant n/\sqrt{k}$, and the running time of algorithm is a polynomial of m, n^r. The algorithm is rather complicated and related with the planted 3SAT model. Further, as parameters suggested the algorithm is not so practical, and we cannot obtain the overwhelming success probability of the algorithm. Still, this theorem gives us an intuition for new direction to construct the fuzzy extractor. We indeed use the algorithm BQ in our last construction, but we do not explicitly describe the algorithm since it is a relatively theoretic construction.

4.2.2 Flipping local functions

In this section we give a description for specific local functions we concern. We consider a slightly general choice induced from a fixed predicate, instead just choose one fixed predicate. Let $P : \{0,1\}^\ell \to \{0,1\}$ be a predicate, and we consider a variant predicate $P_{\boldsymbol{r}} : \{0,1\}^\ell \to \{0,1\}$ that maps $\boldsymbol{x} \mapsto P(\boldsymbol{x} + \boldsymbol{r})$. Now we define a *flipping local function* with a set of vectors $\mathcal{R} = \{\boldsymbol{r}_j\}_{1 \leqslant j \leqslant m}$ by

$$\mathsf{FLF}^m_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{x}) := \left(P(x_{i_{j,1}} + r_{j,1}, \cdots, x_{i_{j,\ell}} + r_{j,\ell})\right)_{1 \leqslant j \leqslant m},$$

along with public parameters $P, \mathcal{I}, \mathcal{R}$. Note that the string $\boldsymbol{r}$ essentially works for random flipping each bit of predicates. We will simply write FLF^m_P by omitting the index sets $\mathcal{I}$ and the flipping vectors set $\mathcal{R}$ if there is no need to specify them, and write $\mathsf{FLF}_{P,\mathcal{I},\mathcal{R}}$ when the length of FLF is obvious in the context.

We remark that while the state-of-the-art analysis on the cryptographic hardness of local random functions have been studied for a fixed single predicate (e.g. [App16, AL18]), many studies had been conducted for more general choice of predicates, for example [CM01, MST06, BR13]. Still, we believe that the flipping local functions enjoy the very similar analysis, and establish the following assumption.

Assumption 4.2.1. *The flipping local function FLF_P is a secure one-way function (PRGs) if the corresponding local function LF_P is a secure one-way function (PRGs, respectively).*

While the main body of this paper—new construction of fuzzy extractor—is written with general predicates, we recommend to reader to keep in mind the recently suggested candidate XOR-MAJ$_{a,b}$ predicate

in [AL18]

$$(x_1 \oplus \cdots \oplus x_a) \oplus \textsc{Maj}_b(x_{a+1}, \cdots, x_{a+b}),$$

where the majority function $\textsc{Maj}_n$ outputs the majority bit of them. For the sake of simplicity, we only consider the case where b is odd. Xor-Maj is considered as a plausible candidate predicate for local PRGs over uniform source and shown to secure against many attacks including a variety class of statistical attack, semi-definite programming, and linear and algebraic attack [OW14, AL18, FPV18]. More concretely the authors of [AL18] suggest local functions with predicate Xor-Maj as a concrete candidate local pseudorandom function (without flipping) for $a \geqslant 2s$ and $b > 16s + 2$ with the stretch $m = n^s$, which rules out all known attacks and their extensions.

Note that, especially for Xor-Maj, the flipping essentially does not affect at all for linear parts, and the remainder non-linear part is a very nontrivial predicate to analyze, at least algebraically.

4.2.3 Noise stability of predicate functions: Xor-Maj

Our key observation for fuzzy extractor construction is that the predicate function P is highly simple so that it sends two close inputs to the same value with an unusual high probability. This notion is formalized by the following definition that says how much stable the function is against (small) perturbation.

Definition 4.2.1. *For $P : \{0,1\}^\ell \to \{0,1\}$ and $t \in [0,1]$, the* noise stability *of P at t is defined by*

$$\mathsf{Stab}_t(P) = 2\Pr[P(\boldsymbol{x}) = P(\boldsymbol{x} + \boldsymbol{\delta})] - 1$$

where $\boldsymbol{x}$ is chosen uniformly over $\{0,1\}^\ell$ and $\boldsymbol{\delta}$ follows the distribution

Ber_t^ℓ.

We have the following proposition about the noise stability of Xor-Maj.

Proposition 4.2.1. *For any $t \in [0,1]$ and non-negative integers a, b, it holds that*

$$\mathsf{Stab}_t(\text{Xor-Maj}_{a,b}) = (1-2t)^a \cdot \mathsf{Stab}_t(\text{Maj}_b).$$

In particular, we have

$$\mathsf{Stab}_t(\text{Xor-Maj}_{a,b}) = O\left((1-2t)^{a+1}\right)$$

Proof. We start from the following lemma.

Lemma 4.2.1. *For any odd integer n and $t \in [0,1]$,*

$$\mathsf{Stab}_t(\text{Maj}_n) = \frac{1}{2^{n-1}} \left(\sum_{h=0}^{\lfloor n/2 \rfloor} \binom{n}{h} \cdot p_{n,h,t} \right). \tag{4.1}$$

where

$$p_{n,h,t} = \sum_{i=0}^{h} \sum_{j=0}^{\lfloor n/2 \rfloor - h + i} \binom{h}{i} \binom{n-h}{j} \cdot t^{i+j} (1-t)^{h-(i+j)}.$$

In particular, the following asymptotic formula holds

$$\left| \mathsf{Stab}_t(\text{Maj}_n) - \frac{2}{\pi} \arcsin(1-2t) \right| = O\left(\frac{1}{\sqrt{nt(1-t)}} \right).$$

Proof. First note that

$$p_{n,h,t} := \Pr_{\boldsymbol{\delta} \leftarrow \mathsf{Ber}_t^n} \left[\text{Maj}(\boldsymbol{x}) = \text{Maj}(\boldsymbol{x} + \boldsymbol{\delta}) \mid \mathsf{HW}(\boldsymbol{x}) = h \right].$$

Then it holds that $p_{n,h,t} = p_{n,n-h,t}$ and

$$\Pr\left[\ \mathsf{HW}(\boldsymbol{x}) = h \mid \boldsymbol{x} \leftarrow \{0,1\}^n\ \right] = \binom{n}{h}/2^n,$$

and hence we have

$$\mathsf{Stab}_t(\textsc{Maj}_n) = \frac{1}{2^{n-1}}\left(\sum_{h=0}^{\lfloor n/2\rfloor}\binom{n}{h}\cdot p_{n,h,t}\right).$$

It only remains to compute $p_{n,h,t}$. For any $\boldsymbol{\delta}$, we denote f_h (f'_h resp) by the number of 0s (1s resp) in $\boldsymbol{x}\oplus\boldsymbol{\delta}$ that was 1s (0s resp) in $\boldsymbol{x}$. Note that

$$\textsc{Maj}_n(\boldsymbol{x}) = \begin{cases} 0 & \text{if } n < 2\mathsf{HW}(\boldsymbol{x}) \\ 1 & \text{otherwise,} \end{cases}$$

and hence $\textsc{Maj}(\boldsymbol{x}\oplus\boldsymbol{\delta})$ remains unchanged if and only if

$$n - 2(h - f_h + f'_h) > 0,$$

or equivalently,

$$f'_h \leqslant \lfloor n/2\rfloor - h + f_h.$$

Thus we have

$$p_{n,h,t} = \left(\sum_{i=0}^{h}\Pr[f_h = i]\cdot\left(\sum_{j=0}^{\lfloor n/2\rfloor - h + i}\Pr[f'_h = j]\right)\right).$$

Since $\boldsymbol{\delta} \leftarrow \mathsf{Ber}_t^n$, we know f_h and f'_h follows $\mathcal{B}(h,t)$ and $\mathcal{B}(n-h,t)$ resp, which completes proof.

The claim for asymptotic behavior is adapted from Sheppard's For-

mula [She99] and Theorem 3.4.2 of [O'D03]. □

Toward the noise stability of XOR-MAJ, we understand it by a bias function of the difference $d_t(f)(\boldsymbol{x}) = f(\boldsymbol{x} + \boldsymbol{\delta}) - f(\boldsymbol{x})$ as follows:

$$\mathsf{Stab}_t(f) = \Pr[d_t(f)(\boldsymbol{x}) = 0] - \Pr[d_t(f)(\boldsymbol{x}) = 1] = -\mathsf{bias}(d_t(f))$$

where a function bias for a binary variable X is defined by

$$\mathsf{bias}(X) := \Pr[X = 1] - \Pr[X = 0].$$

Then the following lemma allows to compute the exact value of stability.

Lemma 4.2.2 (Piling-up lemma [Mat93])**.** *Let $X := \bigoplus_{i=1}^{n} X_i$ for independent binary variables X_i. Then it holds that*

$$\mathsf{bias}(X) = (-1)^{n+1} \prod_{i=1}^{n} \mathsf{bias}(X_i).$$

By Lemma 4.2.2, Lemma 4.2.1 and the relation of bias and stability, the first part is obvious. Moreover, the asymptotic part immediately follows from $\arcsin(x) = O(x)$. □

4.3 From Pseudorandom Local Functions

In this section, we propose a highly simple and intuitive construction of fuzzy extractor for binary alphabet with Hamming distance using the flipping local function. Prior to the beginning, we define the following procedure $\mathsf{SampleRand}$; this definition is totally for readability, because it is just a consecutive uniform sampling.

$\underline{\mathsf{SampleRand}}(n, m, \ell)$:

1. For $j = 1, \cdots, m$:
   (a) Sample a subset I_j of size ℓ uniformly random from $[n]$.
   (b) Sample a uniformly random vector $\boldsymbol{r}_j \leftarrow \{0,1\}^\ell$.
2. Return $\mathcal{I} = \{I_j\}_{1 \leqslant j \leqslant m}, \mathcal{R} = \{\boldsymbol{r}_j\}_{1 \leqslant j \leqslant m}$.

4.3.1 Basic Construction: One-bit Fuzzy Extractor

We start with $(n, 1, t, \delta)$-fuzzy extractor PRGFE_1. Let $n > 0$ be a bit-length of our target bit-string, $t \in [0, 1/2)$ be the target error tolerance ratio, $P : \{0,1\}^\ell \to \{0,1\}$ be a predicate function, and $m > 0$ be an integer that will be specified later, and we write for readability

$$p_t := \frac{\mathsf{Stab}_t(P)}{2}.$$

For $\mathsf{PRGFE}_1.\mathsf{Init}$, we output $\mathsf{pp} = \{m, n, \ell, P\}$, and Figure 4.2 presents $\mathsf{PRGFE}_1.\mathsf{Gen}$ and $\mathsf{PRGFE}_1.\mathsf{Rep}$,

Correctness and Parameters.

We first see the correctness. Let $\boldsymbol{w}$ and $\boldsymbol{w}'$ be n-bit strings such that $\mathsf{HD}(\boldsymbol{w}, \boldsymbol{w}') = t \cdot n$. Since $\mathcal{I}$ and $\mathcal{R}$ are uniformly chosen at random, we know each component of $\mathsf{FLF}_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w})$ and $\mathsf{FLF}_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w}')$ differ with probability $\frac{1}{2} - p_t$. Since each component is independent thanks to the random choice of $\mathcal{R}$, we conclude that

$$\mathsf{HD}\left(G_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w}), G_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w}')\right) \sim \mathcal{B}\left(m, \frac{1}{2} - p_t\right).$$

$\underline{\mathsf{PRGFE}_1.\mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})}$:

1. $b \xleftarrow{\$} \{0,1\}$ // `Extracted bit`
2. $\mathcal{I}, \mathcal{R} \leftarrow \mathsf{SampleRand}(n, m, \ell)$
3. If $b = 1$, $\boldsymbol{h} \leftarrow \mathsf{FLF}_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w})$
4. Else, $\boldsymbol{u} \xleftarrow{\$} \{0,1\}^m$, $\boldsymbol{h} \leftarrow \boldsymbol{u}$
5. $\mathsf{H} \leftarrow (\mathcal{I}, \mathcal{R}, \boldsymbol{h})$ // `Helper`
6. Return (b, H)

$\underline{\mathsf{PRGFE}_1.\mathsf{Rep}(\mathsf{pp}, \boldsymbol{w}', \mathsf{H} = (\mathcal{I}, \mathcal{R}, \boldsymbol{h}))}$:

1. $\boldsymbol{h}' \leftarrow \mathsf{FLF}_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w}')$.
2. $d \leftarrow \mathsf{HD}(\boldsymbol{h}, \boldsymbol{h}')$.
3. If $d \leqslant \frac{m(1-p_t)}{2}$, Return $b' = 1$
4. Else, Return $b' = 0$

Figure 4.2: Basic Fuzzy Extractor

In our scheme view, this corresponds to the case when $\mathsf{PRGFE}_1.\mathsf{Gen}$ extracts $b = 1$. Meanwhile, for the case $b = 0$, the vector $\boldsymbol{h}$ is chosen uniformly at random and hence we have $\mathsf{HD}(\boldsymbol{h}, \boldsymbol{h}') \sim \mathcal{B}(m, \frac{1}{2})$. To sum up, we have

$$\mathsf{HD}(\boldsymbol{h}, \boldsymbol{h}') = \begin{cases} \mathcal{B}(m, \frac{1}{2} - p_t) & \text{if } b = 1 \\ \mathcal{B}(m, \frac{1}{2}) & \text{if } b = 0 \end{cases}.$$

From this the failure probability δ is given by

$$\begin{aligned} \delta = \Pr[b' \neq b] &= \Pr[b = 1] \cdot \Pr\left[X > \frac{m \cdot (1 - p_t)}{2}\right] \\ &\quad + \Pr[b = 0] \cdot \Pr\left[X' \leqslant \frac{m \cdot (1 - p_t)}{2}\right], \end{aligned}$$

where $X \sim \mathcal{B}(m, \frac{1}{2})$, and $X' \sim \mathcal{B}(m, \frac{1}{2} - p_t)$. It is known that for $X \sim$

$\mathcal{B}(n,t)$, it holds that

$$\Pr[X \leqslant a] \leqslant \exp\left(-2 \cdot \frac{(nt-a)^2}{n}\right)$$

named Hoeffding's bound, and we conclude

$$\delta \leqslant \exp\left(-\frac{m \cdot p_t^2}{2}\right). \tag{4.2}$$

The helper consists of $\ell \cdot m$ numbers of indices in $[n]$ (corresponding to $\mathcal{I}$) and $\ell \cdot m$ bits string (corresponding to $\mathcal{R}$), and m bits string (corresponding to $\boldsymbol{h}$) for one extracted bit. Thus the total helper size would be

$$|\mathsf{H}| = (\ell\lceil \log n \rceil + \ell + 1) \cdot m \tag{4.3}$$

bits.

Remark. The most part of helper in our scheme consists of random bit strings $\mathcal{I}$ and $\mathcal{R}$. This can be reduced by a short seed of appropriate random number generator, as in many LWE-based cryptosystems [ADPS16, Gal13, CMNT11], and then the helper size would be only m. We note that this is secure in the random oracle model, for example see [Gal13].

4.3.2 Expansion to multi-bit Fuzzy Extractor

We consider two natural expansion of PRGFE_1 to κ-bit length key extraction scheme, say PRGFE^1_κ and PRGFE^2_κ. The basic idea for both is to simply call κ times of PRGFE_1, but the input bit-string $\boldsymbol{w}$ is differently fed to PRGFE_1. We will show the both constructions enjoy reusable security, but over different type of random sources later.

The second expansion PRGFE^2_κ is designed for the case where the input

$\underline{\mathsf{PRGFE}^1_\kappa.\mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})}$:

1. For $i = 1, \cdots, \kappa$:
   (a) $(\mathsf{H}_i, b_i) \leftarrow \mathsf{PRGFE}_1.\mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})$
2. $\boldsymbol{b} \leftarrow (b_1, \cdots, b_\kappa)$
3. Return $(\mathsf{H} = \{\mathsf{H}_i\}_{1 \leqslant i \leqslant \kappa}, \boldsymbol{b})$

$\underline{\mathsf{PRGFE}^1_\kappa.\mathsf{Rep}(\mathsf{pp}, \boldsymbol{w}', \mathsf{H} = \{\mathsf{H}_i\}_{1 \leqslant i \leqslant \kappa})}$:

1. For $i = 1, \cdots, \kappa$:
   (a) $b'_i \leftarrow \mathsf{PRGFE}_1.\mathsf{Rep}(\mathsf{pp}, \boldsymbol{w}, \mathsf{H}_i)$
2. $\boldsymbol{b}' \leftarrow (b'_1, \cdots, b'_\kappa)$
3. Return $\boldsymbol{b}'$

Figure 4.3: PRGFE^1_κ: First expansion of κ-bit Fuzzy Extractor

string $\boldsymbol{w} \in \{0,1\}^n$ can be divided into κ mutually independent blocks having length n_i. First we define $\mathsf{PRGFE}^2_\kappa.\mathsf{Init}$ outputs $\mathsf{pp} = \{\mathsf{pp}_i\}$ where $\mathsf{pp}_i = \{m, n_i, \ell, P\}$.

$\underline{\mathsf{PRGFE}^2_\kappa.\mathsf{Gen}(\mathsf{pp}, \boldsymbol{w})}$:

1. Parse $\boldsymbol{w}$ into $\boldsymbol{w}_i \in \{0,1\}^{n_i}$ $(i \in [\kappa])$
2. For $i = 1, \cdots, \kappa$:
   (a) $(\mathsf{H}_i, b_i) \leftarrow \mathsf{PRGFE}_1.\mathsf{Gen}(\mathsf{pp}_i, \boldsymbol{w}_i)$
3. $\boldsymbol{b} \leftarrow (b_1, \cdots, b_\kappa)$
4. Return $(\mathsf{H} = \{\mathsf{H}_i\}_{1 \leqslant i \leqslant \kappa}, \boldsymbol{b})$

$\underline{\mathsf{PRGFE}^2_\kappa.\mathsf{Rep}(\mathsf{pp}, \boldsymbol{w}', \mathsf{H} = \{\mathsf{H}_i\}_{1 \leqslant i \leqslant \kappa})}$:

1. Parse $\boldsymbol{w}$ into $\boldsymbol{w}_i \in \{0,1\}^{n_i}$ $(i \in [\kappa])$
2. For $i = 1, \cdots, \kappa$:
   (a) $b'_i \leftarrow \mathsf{PRGFE}_\kappa.\mathsf{Rep}(\mathsf{pp}_i, \boldsymbol{w}_i, \mathsf{H}_i)$
3. $\boldsymbol{b}' \leftarrow (b'_1, \cdots, b'_\kappa)$
4. Return $\boldsymbol{b}'$

Figure 4.4: PRGFE^2_κ: Second expansion of κ-bit Fuzzy Extractor:

Correctness and Helper size.

We deal with the correctness, helper size and running times for both constructions at once. For both of constructions, we require every bit b_i be

correctly reproduced and hence from (4.2) we bound the failure probability δ as

$$\begin{aligned}\delta &\leqslant \left(1-\left(1-\exp\left(-\frac{m\cdot p_t^2}{2}\right)\right)^{\kappa}\right)\\ &\leqslant \kappa\exp\left(-\frac{m\cdot p_t^2}{2}\right)=\kappa\exp\left(-\frac{m\cdot \mathsf{Stab}(P)_t^2}{8}\right).\end{aligned} \tag{4.4}$$

Equivalently, we can say that the parameter m for obtaining δ should be

$$m \geqslant \frac{8\ln\left(\kappa\cdot\delta^{-1}\right)}{\mathsf{Stab}_t(P)^2}. \tag{4.5}$$

The total helper size would be κ times of (4.3), namely

$$|\mathsf{H}| = \kappa\cdot(\ell\cdot(\lceil\log n\rceil+1)+1)\cdot m,$$

which can be compressed into $|\mathsf{H}'| = \kappa m$ as in Remark 4.3.1.

We note that, the parameter m determines the overall performance; helper size and algorithm running times. So far we only consider the correctness part to derive one condition for m by (4.3). In the next section we obtain one more condition for m from the security requirement, which enables us to determine m.

4.3.3 Indistinguishable Reusability

We first state the following main theorem that says for any random source W where $\mathsf{FLF}_P(W)$ is pseudorandom, PRGFE_κ^1 is IND-reusable. Recall that, for a predicate P where $\mathsf{LF}_P(\mathcal{U}_n)$ is pseudorandom, we plausibly assume that $\mathsf{FLF}_P(\mathcal{U}_n)$ for $n = O(\lambda)$ is also pseudorandom, which provides a concrete example of random source $W = \mathcal{U}_n$ where PRGFE_1 is assumed to be

IND-reusable.

Theorem 4.3.1 (IND-reusability of PRGFE^1_κ)**.** *Let P be a predicate function. Let $\mathcal{W}$ be a family of distributions W over $\{0,1\}^n$ such that the flipping local function FLF_P is a ε-PRG over W with stretch $(\rho+1)\kappa m$. Then PRGFE^1_κ in Figure 4.3 is a (ρ,ε)-IND-reusable (n,κ,t,δ)-fuzzy extractor over $\mathcal{W}$ where failure probability is bounded by*

$$\delta \leqslant \kappa \exp\left(-m \cdot \mathsf{Stab}_t(P)^2/8\right).$$

Proof. The failure probability is already known by (4.4). Let $W \in \mathcal{W}$ be a random source. For the proof of reusability, we consider a series of experiments Exp_0, Exp_1, and Exp_2, where $\mathsf{Exp}_0 = \mathsf{Exp}^W_{\mathsf{IND-reu}}$.

Exp_0: This is exactly the reusability experiment, and in Figure 4.5 gives the detailed process in terms of PRGFE_1.

Exp_1: We change step (c) of $\mathsf{Chal}(\boldsymbol{\delta}^{(k)})$ to compute the local function $\mathsf{FLF}^m_{P,\mathcal{I},\mathcal{R}}$ on input $\boldsymbol{w}$, instead of $\boldsymbol{w}+\boldsymbol{\delta}^{(k)}$; see step (c) of $\mathsf{Chal}(\boldsymbol{\delta}^{(k)})$ of Figure 4.5.

Claim 1. Any adversary $\mathcal{A}$ winning Exp_1 with some probability implies an adversary $\mathcal{B}$ winning Exp_0 with the same probability, and vice versa.

Proof of Claim. On Exp_0, $\mathcal{A}$ learns from $\boldsymbol{\delta}^{(k)}$ queries

$$\Omega_0 = \left\{ \boldsymbol{\delta}^{(k)}, \boldsymbol{b}^{(k)}, \left(\mathcal{I}_i^{(k)}, \mathcal{R}_i^{(k)}, \boldsymbol{h}_i^{(k)}\right)_{1\leqslant i\leqslant\kappa} \right\}_{1\leqslant k\leqslant\rho}$$

where

$$\boldsymbol{h}_i^{(k)} = \begin{cases} \mathsf{FLF}_{P,\mathcal{I}_i^{(k)},\mathcal{R}_i^{(k)}}(\boldsymbol{w}+\boldsymbol{\delta}^{(k)}) & \text{if } b_i^{(k)}=1 \\ \boldsymbol{u}_i^{(k)} & \text{if } b_i^{(k)}=0. \end{cases}$$

$\underline{\mathsf{Init}(\mathsf{pp})}$: $\boxed{\mathsf{Exp}_0}$, $\boxed{\mathsf{Exp}_1}$, $\boxed{\mathsf{Exp}_2}$

1. $\boldsymbol{w} \leftarrow W$
2. For $i = 1, \cdots, \kappa$:
   (a) $b_i \xleftarrow{\$} \{0,1\}$
   (b) $\mathcal{I}_i, \mathcal{R}_i \leftarrow \mathsf{SampleRand}(n, m, \ell)$
   (c) $\boxed{\text{If } b_i = 1,\ \boldsymbol{h}_i \leftarrow \mathsf{FLF}^m_{P,\mathcal{I}_i,\mathcal{R}_i}(\boldsymbol{w}) \text{ Else, } \boldsymbol{u}_i \xleftarrow{\$} \{0,1\}^m, \boldsymbol{h}_i \leftarrow \boldsymbol{u}}$
   $\boxed{\text{Regardless of } b_i,\ \boldsymbol{u}_i \xleftarrow{\$} \{0,1\}^m, \boldsymbol{h}_i \leftarrow \boldsymbol{u}}$
   (d) $\mathsf{H}_i \leftarrow (\mathcal{I}_i, \mathcal{R}_i, \boldsymbol{h}_i)$
3. $\boldsymbol{b} \leftarrow (b_1, \cdots, b_\kappa)$
4. $\mathsf{H} = \{\mathsf{H}_i\}_{1 \leqslant i \leqslant \kappa}$
5. $\beta \xleftarrow{\$} \{0,1\}$
6. If $\beta = 1$, Return $(\boldsymbol{b}, \mathsf{H})$
   Else, $\boldsymbol{u} \xleftarrow{\$} \{0,1\}^\kappa$, Return $(\boldsymbol{u}, \mathsf{H})$

$\underline{\mathsf{Chal}(\boldsymbol{\delta}^{(k)})}$: $\boxed{\mathsf{Exp}_0}$, $\boxed{\mathsf{Exp}_1}$, $\boxed{\mathsf{Exp}_2}$

1. For $i = 1, \cdots, \kappa$:
   (a) $b_i^{(k)} \xleftarrow{\$} \{0,1\}$
   (b) $\mathcal{I}_i^{(k)}, \mathcal{R}_i^{(k)} \leftarrow \mathsf{SampleRand}(n, m, \ell)$
   (c) $\boxed{\text{If } b_i^{(k)} = 1,\ \boldsymbol{h}_i^{(k)} \leftarrow \mathsf{FLF}^m_{P,\mathcal{I}_i^{(k)},\mathcal{R}_i^{(k)}}(\boldsymbol{w} + \boldsymbol{\delta}^{(k)}) \text{ Else, } \boldsymbol{u}_i^{(k)} \xleftarrow{\$} \{0,1\}^m, \boldsymbol{h}_i^{(k)} \leftarrow \boldsymbol{u}}$
   $\boxed{\boldsymbol{h}_i^{(k)} \leftarrow \mathsf{FLF}^m_{P,\mathcal{I}_i^{(k)},\mathcal{R}_i^{(k)}}(\boldsymbol{w})}$
   $\boxed{\text{Regardless of } b_i^{(k)},\ \boldsymbol{u}_i^{(k)} \xleftarrow{\$} \{0,1\}^m, \boldsymbol{h}_i^{(k)} \leftarrow \boldsymbol{u}}$
   (d) $\mathsf{H}_i^{(k)} \leftarrow (\mathcal{I}_i^{(k)}, \mathcal{R}_i^{(k)}, \boldsymbol{h}_i^{(k)})$
2. $\boldsymbol{b}^{(k)} \leftarrow (b_1^{(k)}, \cdots, b_\kappa^{(k)})$
3. $\mathsf{H}^{(k)} = \{\mathsf{H}_i^{(k)}\}_{1 \leqslant i \leqslant \kappa}$
4. Return $(b^{(k)}, \mathsf{H}^{(k)})$

$\underline{\mathsf{Fin}(\beta')}$:

1. If $\beta = \beta'$, Return 1
   Else, Return 0

Figure 4.5: Overview of proof

Here, from the definition of FLF we know that

$$\mathsf{FLF}_{P,\mathcal{I}_i^{(k)},\mathcal{R}_i^{(k)}}(\boldsymbol{w} + \boldsymbol{\delta}^{(k)}) = \mathsf{FLF}_{P,\mathcal{I}_i^{(k)},\mathcal{R}_i^{(k)}+\boldsymbol{\delta}^{(k)}}(\boldsymbol{w}),$$

and since $\mathcal{A}$ knows $\boldsymbol{\delta}^{(k)}$, the knowledge of Ω_0 implies to the knowledge of

$$\Omega_1 = \left\{ \boldsymbol{\delta}^{(k)}, \boldsymbol{b}^{(k)}, \left(\mathcal{I}_i^{(k)}, \mathcal{R}_i^{(k)} + \boldsymbol{\delta}^{(k)}, \boldsymbol{h}_i^{(k)} \right)_{1 \leqslant i \leqslant \kappa} \right\}_{1 \leqslant k \leqslant \rho}$$

where

$$\boldsymbol{h}_i^{(k)} = \begin{cases} \mathsf{FLF}_{P, \mathcal{I}_i^{(k)}, \mathcal{R}_i^{(k)} + \boldsymbol{\delta}^{(k)}}(\boldsymbol{w}) & \text{if } b_i^{(k)} = 1 \\ \boldsymbol{u}_i^{(k)} & \text{if } b_i^{(k)} = 0. \end{cases}$$

Thanks to the uniformly random choice of $\mathcal{R}_i^{(k)}$, Ω_1 is actually the view of an adversary $\mathcal{B}$ playing Exp_1, from which we prove the claim. □

Exp_2: We now change step (c) of Init and Chal to output $\boldsymbol{h}_i$ and $\boldsymbol{h}_i^{(k)}$ as a uniform string regardless of the choice of extracted bit b and b_k. Since the adversary's view in Exp_2 is exactly same regardless of β, we have $\Pr[\mathsf{Exp}_2 = 1] = 1/2$.

Claim 2. $|\Pr[\mathsf{Exp}_1 = 1] - \Pr[\mathsf{Exp}_2 = 1]| \leqslant \varepsilon$

Proof of Claim. During the execution of Exp_1, the adversary obtains at most $(\rho+1)\kappa m$ bits of FLF_P outputs; $\rho\kappa m$ bits from Chal phases, and κm bits from Init phase. Since the other view of adversary is same for both experiments, an adversary that distinguishes those experiments can be used to distinguish less than $(\rho+1)\kappa m$-length FLF_P outputs from uniform, whose advantage is bounded by ε from the pseudorandomness of flipping local functions. □

By combining Claim 1 and Claim 2, we reach conclusion.

□

4.3.4 One-way Reusability

We show in the previous section that PRGFE^1_κ is secure for any source implying pseudorandomness of underlying flipping local functions. However, relying on only the strong PRG assumption would lead to restrictive use of our idea, since it is still obscure that whether one's interest random source satisfies such assumption. In this regard, we show that PRGFE^2_κ achieves one-wayness based reusable security over some random sources that requires quite different condition of for FLF_P. In fact, we consider random sources that consist of several independent blocks W_i where the flipping local function over W_i is ε-PRG with $\varepsilon \leqslant 1/2$.

Theorem 4.3.2 (OW-reusability of PRGFE^2_κ)**.** *Let P be a predicate function. Let $\mathcal{W}$ be a family of distribution W over $\{0,1\}^n$ of the form $W = (W_1, \cdots, W_\kappa)$ such that*

- *W_is are mutually independent.*
- *FLF_P is an ε-PRG with $\varepsilon < 1/2$ with stretch $(\rho+1)m$ for every W_i.*

Then PRGFE^2_κ in Figure 4.4 is a (ρ, ε')-OW-reusable (n, κ, t, δ)-fuzzy extractor over $\mathcal{W}$ where

$$\varepsilon' \leqslant \left(\frac{1}{2} + \varepsilon\right)^\kappa \text{ and } \delta \leqslant \kappa \exp\left(-m \cdot \mathsf{Stab}_t(P)^2/8\right).$$

Proof. Note that for one-bit extraction fuzzy extractor, (ρ, ε)-IND-reusable security is equivalent to $(\rho, 1/2+\varepsilon)$-OW-reusable security. Then, Theorem 4.3.1 applied for $\kappa = 1$ for each W_i implies, PRGFE^1_1 is $(\rho, 1/2+\varepsilon)$-OW-reusable over W_i. From the mutually independence assumption on W_i, PRGFE^2_κ can be understood by a κ concatenation of *independent* PRGFE^1_1,

and hence we conclude that inverting whole κ-bit succeeds with probability $\leqslant \left(\frac{1}{2} + \varepsilon\right)^{\kappa}$. □

Remark. We can convert this OW-reusable fuzzy extractor into IND-reusable one in random oracle model, using generic conversion due to [ACEK17]. This can be simply done by letting the extracted key by $H(\boldsymbol{b})$ for a function H modeled by random oracle; since H is random oracle, the only strategy of the adversary distinguishing $H(\boldsymbol{b})$ and uniform string $\boldsymbol{u}$ is to find the preimage $\boldsymbol{b}$.

Instantiation

We show the efficiency of our constructions by instantiating with Xor-Maj predicate. Clearly, the size of helper and running time of algorithms Gen and Rep is proportional to m, and we argue that m is in polynomial of other parameters like ρ, n, δ, and especially of error ratio t. We will assume here t is in $[0, 0.5 - \nu]$ for some constant $\nu > 0$, which suffices for fuzzy extractor.

For that, we first recall that from [AL18], the local function of Xor-Maj$_{a,b}$ where $a = \lceil 2s \rceil$ and $b = \lfloor 16s + 2.5 \rceil$ is pseudorandom over $\mathcal{U}_n$ until the stretch n^s, which yields $\mathsf{Stab}_t(P) = O\left((1-2t)^{2s+1}\right) = O\left((1-2t)^{2s}\right)$ from Proposition 4.2.1. Based on this fact, we assume that our target random source W yields the local function of Xor-Maj$_{a,b}$ with $a = cs$ for some $c \geqslant 2$ and a corresponding b is a ε-PRG over W with stretch n^s, which gives $\mathsf{Stab}_t(P) = O\left((1-2t)^{cs}\right)$.

Now we summarize the conditions for m and s that gives ρ-IND-reusable (n, κ, t, δ)-fuzzy extractor from Theorem 4.3.1.

- For failure probability δ, we set $m = O\left(\dfrac{\ln\left(\kappa \cdot \delta^{-1}\right)}{(1-2t)^{2cs}}\right)$.

- For ρ-reusability, we choose s such that $n^s \geqslant (\rho+1)\kappa m$ for PRGFE^1_κ.

By combining two conditions, we reach one condition on s by

$$s = O\left(\frac{\log\left(\rho\kappa \cdot \ln\left(\kappa \cdot \delta^{-1}\right)\right)}{\log n - 2c \cdot \log(1-2t)}\right),$$

which implies

$$s = O\left(\frac{\log\left(\rho\kappa \cdot \ln\left(\kappa \cdot \delta^{-1}\right)\right)}{\log n - 2c}\right)$$

since $\log(1-2t) = O(1)$ in our interest range of t. Finally, we conclude

$$m = O\left(\frac{\ln\left(\kappa \cdot \delta^{-1}\right)}{(1-2t)^{2cs}}\right) \leqslant O\left(\ln\left(\kappa \cdot \delta^{-1}\right) \cdot t^{2cs}\right).$$

since $\frac{1}{1-2t} = O(t)$ in our interest range of t. To sum up, our helper size and the running time of all algorithms are polynomials in t, which is the first achievement for non-sketch-and-extract schemes.

We finally remark that this analysis contains quite a few hidden constants, and some non-tight approximations, for instance $\log(1-2t) = O(1)$ and $\frac{1}{1-2t} = O(t)$, and hence the actual parameter setting would be smaller so that results in much efficient instantiation.

Remark. The previous best result [CFP+16] had helper size and reproduce time exponential in t. To be precise, to achieve ρ-reusability and failure probability δ, it publishes $\ln(\delta^{-1}) \cdot \exp(t\lambda)$ numbers of digital lockers as helper. Moreover, as generate (reproduce, resp) phase need to generate (unlock, resp) every digital lockers, the running time is also proportional to $\ln(\delta^{-1}) \cdot \exp(t\lambda)$.

[†]For brevity, we only consider PRGFE^1_κ case. For PRGFE^2_κ we need $n^s \geqslant (\rho+1)m$.

4.4 From Local One-way Functions

The main drawback of the pseudorandomness-based constructions of the previous section is that the condition for random source is harsh, and may not apply for the noisy sources in real world: Although the second construction requires somewhat mild pseudorandomness condition for each block, the assumption that the random source consists of independent blocks is still uncomfortable.

In this section, we give another polynomial-time construction that achieves the reusable security over random source yielding one-way local function. As we discussed in Section 4.2, the one-wayness of local function on non-uniform source is quite plausible.

The main idea is very simple; encode the input $\boldsymbol{w}$ using local functions in the generation phase and also make a random string $\boldsymbol{r}$; the extracted string would be $\boldsymbol{b} = H(\boldsymbol{w}||\boldsymbol{r})$ for a hash function H modelled by random oracle. In the reproduce phase, recover the original input $\boldsymbol{w}$ by invoking the algorithm BQ in Theorem 4.2.1 with the advice $\boldsymbol{w}'$, a noisy input, and then recover $\boldsymbol{r}$.

We remark that this construction is exactly the same to the secure sketch based fuzzy extractor, and here the local function plays the role of secure sketch. The difference is, we only care about the one-wayness here whereas the usual discussion of this construction require the secure sketch to have a constraint on the min-entropy. Our result below can be generally understood as (reusable) one-way secure sketch leads to (reusable) one-way fuzzy extractor in the random oracle model.[‡]

Now we formally describe our one-way fuzzy extractor as in Figure 4.6.

[‡]We remark that since we use the one-wayness as security notion, we cannot weaken the requirement of random oracles to randomness extractors.

Here $H : \{0,1\}^* \to \{0,1\}^\kappa$ be a hash function modelled by random oracle.

$\underline{\mathsf{OWFE.Gen}(\mathsf{pp}, \boldsymbol{w})}$:

1. $\mathcal{I}, \mathcal{R} \leftarrow \mathsf{SampleRand}(n, m, \ell)$
2. $\boldsymbol{h}_1 \leftarrow \mathsf{FLF}_{P,\mathcal{I},\mathcal{R}}(\boldsymbol{w})$, $\boldsymbol{h}_2 \xleftarrow{\$} \{0,1\}^\lambda$
3. $\mathsf{H} \leftarrow (\mathcal{I}, \mathcal{R}, \boldsymbol{h}_1, \boldsymbol{h}_2)$ // `Helper`
4. $\boldsymbol{b} \leftarrow H(\boldsymbol{w}||\boldsymbol{h}_2)$ // `Extracted bits`
5. Return $(\boldsymbol{b}, \mathsf{H})$

$\underline{\mathsf{OWFE.Rep}(\mathsf{pp}, \boldsymbol{w}', \mathsf{H} = (\mathcal{I}, \mathcal{R}, \boldsymbol{h}_1, \boldsymbol{h}_2))}$:

1. $\boldsymbol{w} \leftarrow \mathsf{BQ}(\boldsymbol{h}_1, \boldsymbol{w}')$
2. $\boldsymbol{b} \leftarrow H(\boldsymbol{w}||\boldsymbol{h}_2)$

Figure 4.6: One-way Fuzzy Extractor

We discuss the reusable security and correctness. Unfortunately, this scheme has $o(1)$ error probability which cannot be negligible, and for the smaller error probability ($O(n^{-r})$) one has to spend longer time complexity (polynomial of n^r). The reusable security follows from the one-wayness of local functions and the random oracle model.

Theorem 4.4.1 (OW-reusability of OWFE)**.** *Let $P : \{0,1\}^\ell \to \{0,1\}$ be a predicate function. Let $\mathcal{W}$ be a family of distributions W over $\{0,1\}^n$ such that the flipping local function FLF_P is a ε-OW over $\mathcal{W}$ with stretch $(\rho+1)m$. If $m \geqslant n \cdot (k/\mu)^{2\ell}$ for $\mu = 1/2 - t$ for the universal constant k in Theorem 4.2.1, then OWFE in Figure 4.6 is a (ρ, ε)-OW-reusable (n, κ, t, δ)-fuzzy extractor over $\mathcal{W}$ where failure probability δ is bounded by $o(1)$.*

Proof. The correctness is directly derived from Theorem 4.2.1. We argue about the reusable security part. By following the conversion from Exp_0 to

Exp_1 in the proof of Theorem 4.3.1, we may assume that every Chal query is done by $\boldsymbol{\delta}^{(i)} = \mathbf{0}$. Note that for every helper $\mathsf{H}^{(i)} = (\mathcal{I}^{(i)}, \mathcal{R}^{(i)}, \boldsymbol{h}_1^{(i)}, \boldsymbol{h}_2^{(i)})$ and key $\boldsymbol{b}^{(i)} = H(\boldsymbol{w}||\boldsymbol{h}_2^{(i)})$ pair that the adversary obtain during experiment, only meaningful information would be $(\mathcal{I}^{(i)}, \mathcal{R}^{(i)}, \boldsymbol{h}_1^{(i)})$ since H is a random oracle. Then, the adversary is asked to find $\boldsymbol{b} = H(\boldsymbol{w}||\boldsymbol{h}_2)$ from at most $(\rho+1)m$ length of FLF_P outputs. Again, since H is a random oracle, the only strategy of adversary is to find the preimage $(\boldsymbol{w}||\boldsymbol{h}_2)$ of $\boldsymbol{b}$, whose probability is bounded by ε by the one-wayness of FLF_P. □

Instantiation

Now we discuss the asymptotic efficiency of our construction. We assume that the bit length n of source is sufficiently large, and that there is a family of local functions with locality ℓ that achieves the one-way security over $\mathcal{W}$ and has a stretch $n^{c\ell}$ for constant c[§]. In particular, Xor-Maj is conjectured to satisfy the stretch $n^{(\ell-3)/18}$ with one-wayness. Also, we assume that the error rate $t < 1/2$ is an *arbitrary* positive constant, i.e. $\mu > 0$ is an arbitrary fixed constant. Then, the possible number of reusability is

$$\frac{n^{c\ell}}{n \cdot (k/\mu)^{2\ell}} = \left(\frac{\mu n^c}{k}\right)^{\ell} / n.$$

Thus we obtain the following proposition.

Proposition 4.4.1. *Let* $\mathsf{LF} : \{0,1\}^n \to \{0,1\}^{n^s}$ *be a local function with predicate* Xor-Maj *with locality* ℓ *over* $\mathcal{W}$*. Assuming that* LF *is a one-way function over* $\mathcal{W}$*, there is a one-way fuzzy extractor with reusability* $\Omega(\mu^{\ell} n^{s-1})$*, error probability* $o(1)$ *and the running time* $\mathsf{poly}(n \cdot (1/\mu)^{2\ell})$ *for error rate* $t = 1/2 - \mu$ *of source.*

[§]All promising candidate local functions satisfy this asymptotic.

Note that the helper size is as same as the PRG-based construction, and the running time is also polynomial in t.

Chapter 5

Concrete Security of Homomorphic Encryption

In this chapter, upon the current dual attack framework, we apply MitM attacks for LWE instead of exhaustive search. For that, we first observe that Odlyzko's MitM attack on NTRU [HGSW03] can be easily adapted to the literature of LWE, and we give an explicit algorithm and rigorous analysis for it. The cost of this attack is proportional to the square root of the number of candidate secret vector, while it is less sensitive to the absolute size of error when the ratio of error and modulus is sufficiently small. Thus, this MitM attack is highly appropriate for the trade-offed LWE sample for the large modulus case and from this observation,

From this observation, we propose a new hybrid attack of the dual attack and MitM attack. Our hybrid attack shows significant performance improvement on the sparse ternary secret LWE problems, which are used in two homomorphic encryptions HElib [HS14] and HEAAN [CKKS17]*.

*SEAL also needs to use the sparse ternary key to support the bootstrapping.

We estimate our attack complexity for several parameters that are in the currently used parameter range for the HEs. This result shows that our attack can solve the sparse ternary secret LWE problems in more than 1000 times faster compared to the previous attacks on average.

5.1 Albrecht's Improved Dual Attack

In this section, we give a detailed descrption of the dual attack and its recent variant suggested by Albrecht [Alb17], which is known as the best attack on the underlying LWE problems of fully homomorphic encryptions.

5.1.1 Simple Dual Lattice Attack

The dual lattice attack is an algorithm to solve LWE. The main idea of the dual attack is to exploit a short vector in the following orthogonal lattice

$$\Lambda_q^{\perp}(A) = \{\boldsymbol{v} \in \mathbb{Z}^n : \boldsymbol{v}^t A \equiv_q \mathbf{0}\}.$$

More precisely, for a short vector $\boldsymbol{y}$ in $\Lambda_q^{\perp}$ and an LWE sample $(A, \boldsymbol{b})$, one has

$$\langle \boldsymbol{y}, \boldsymbol{b} \rangle = \langle \boldsymbol{y}, A\boldsymbol{s} + \boldsymbol{e} \rangle = \langle \boldsymbol{y}, A\boldsymbol{s} \rangle + \langle \boldsymbol{y}, \boldsymbol{e} \rangle \equiv_q \langle \boldsymbol{y}, \boldsymbol{e} \rangle$$

and this yields $[\langle \boldsymbol{y}, \boldsymbol{b} \rangle]_q = \langle \boldsymbol{y}, \boldsymbol{e} \rangle$, which is significantly shorter than q. On the other hand, if the given sample $(A, \boldsymbol{b})$ is uniform random then $[\langle \boldsymbol{y}, \boldsymbol{b} \rangle]_q$ is a random value which is not small compared to the previous case. By applying this procedure for different $\boldsymbol{y}$'s, we obtain the distinguishing algorithm with overwhelming success probability. Thus we can solve the LWE problem using the smallness of this inner product.

See [CH18].

For LWE cases with small secrets, a natural improvement of dual attack can be obtained by considering the scaled or normal form of dual lattice. More precisely, the scaled normal dual lattice is defined by

$$\Lambda_{q,c}(A) = \{(\boldsymbol{v}_1, \boldsymbol{v}_2) \in \mathbb{Z}^m \times (\tfrac{1}{c}\mathbb{Z})^n : \boldsymbol{v}_1^t A \equiv_q c \cdot \boldsymbol{v}_2\}.$$

As in the dual attack, we find a short vector $(\boldsymbol{y}_1, \boldsymbol{y}_2) \in \Lambda_{q,c}(A)$ and then compute the inner product as follows

$$\langle \boldsymbol{y}_1, \boldsymbol{b} \rangle = \langle \boldsymbol{y}_1, A\boldsymbol{s} \rangle + \langle \boldsymbol{y}_1, \boldsymbol{e} \rangle \equiv_q c \cdot \langle \boldsymbol{y}_2, \boldsymbol{s} \rangle + \langle \boldsymbol{y}_1, \boldsymbol{e} \rangle$$

for the LWE sample $(A, \boldsymbol{b} = A\boldsymbol{s} + \boldsymbol{e})$ that allows us to solve the DLWE problem.

Choice of c.

We take the constant c to satisfy $|c \cdot \langle \boldsymbol{y}_2, \boldsymbol{s} \rangle| \approx \mathrm{E}[|\langle \boldsymbol{y}_1, \boldsymbol{e} \rangle|]$, in order that each summand equally contributes to error e. First we estimate $\mathrm{E}[|\langle \boldsymbol{y}_1, \boldsymbol{e} \rangle|] \approx \frac{\alpha q}{\sqrt{2\pi}} \cdot \|\boldsymbol{y}_1\|$, and then c would be taken to satisfy

$$c \approx \frac{\alpha q}{\sqrt{2\pi}} \cdot \frac{\|\boldsymbol{y}_1\|}{|\langle \boldsymbol{y}_2, \boldsymbol{s} \rangle|}.$$

Although we assume that $\boldsymbol{y}$ is short, Since the exact size of $\boldsymbol{y}_1$ and $\langle \boldsymbol{y}_2, \boldsymbol{s} \rangle$ are not sure, we heuristically assume that $\|\boldsymbol{y}_1\| \approx \sqrt{\frac{m}{m+n}}\|\boldsymbol{y}\|$ and $|\langle \boldsymbol{y}_2, \boldsymbol{s} \rangle| \approx \sqrt{\frac{h}{m+n}}\|\boldsymbol{y}\|$.

Assumption 5.1.1. *Let $\boldsymbol{y} \in L_c(A)$ be a short vector obtained from lattice reduction. Then each entry of $\boldsymbol{y}$ has similar size $\|\boldsymbol{y}\|/\sqrt{m+n}$.*

5.1.2 Improved Dual Attack

Now we review the improvement on the dual attack on the sparse secret LWE problem [Alb17]. Most of the techniques described in this section are applicable to our hybrid attack. Hereafter we assume that the secret key $\boldsymbol{s}$ is in $\mathcal{B}_{n,h}$ for some $h \ll n$.

Assumption on $\boldsymbol{s}$.

To exploit the sparsity of secret key, Albrecht suggests to solve the LWE problem by dual lattice attack with the assumption that some coordinates of secret key are zero. More precisely, parse the matrix A into $A_1||A_2$ for two matrix $A_1 \in \mathbb{Z}_q^{m\times(n-k)}$ and $A_2 \in \mathbb{Z}_q^{m\times k}$. If the part of secret key that corresponds to A_2 is the zero vector, Then it holds that $\boldsymbol{b} = A\boldsymbol{s} + \boldsymbol{e} = A_1\boldsymbol{s}_1 + \boldsymbol{e}$, for the parsed secret key $\boldsymbol{s} = (\boldsymbol{s}_1||\boldsymbol{s}_2) \in \mathbb{Z}_q^{n-k} \times \mathbb{Z}_q^k$ such that $\boldsymbol{s}_2 = \boldsymbol{0}$. Thus the dual attack on A_1 using $(\boldsymbol{y}_1, \boldsymbol{y}_2) \in \Lambda_{q,c}(A_1)$ proceeds

$$\begin{aligned}\langle \boldsymbol{y}_1, \boldsymbol{b}\rangle &= \langle \boldsymbol{y}_1, A_1\boldsymbol{s}_1 + \boldsymbol{e}\rangle \\ &= \langle \boldsymbol{y}_1, A_1\boldsymbol{s}_1\rangle + \langle \boldsymbol{y}_1, \boldsymbol{e}\rangle \\ &\equiv_q c \cdot \langle \boldsymbol{y}_2, \boldsymbol{s}_1\rangle + \langle \boldsymbol{y}_1, \boldsymbol{e}\rangle.\end{aligned}$$

Since it is sufficient to run the lattice reduction algorithm in dimension $n-k$ instead of n, this assumption yields the faster time to solve the DLWE problem. The drawback is the probability that the assumption holds; we minimize the product of the inverse of the probability and the time complexity to solve DLWE with this assumption by choosing appropriate k.

Relaxed Assumption.

Albrecht introduces another method to relax the assumption. When $\boldsymbol{s}_2 \neq \mathbf{0}$, the dual attack on A_1 yields

$$\begin{aligned}\langle \boldsymbol{y}_1, \boldsymbol{b} \rangle &= \langle \boldsymbol{y}_1, A_1 \boldsymbol{s}_1 \rangle + \langle \boldsymbol{y}_1, A_2 \boldsymbol{s}_2 \rangle + \langle \boldsymbol{y}_1, \boldsymbol{e} \rangle \\ &\equiv_q \boldsymbol{y}_1^t A_2 \boldsymbol{s}_2 + c \cdot \boldsymbol{y}_2^t \boldsymbol{s}_1 + \boldsymbol{y}_1^t \boldsymbol{e}\end{aligned}$$

and $c \cdot \boldsymbol{y}_2^t \boldsymbol{s}_1 + \boldsymbol{y}_1^t \boldsymbol{e}$ is relatively small when the sample is from LWE. We assume that the coordinates of $\boldsymbol{s}_2$ are all but up to h' zero, instead of zero vector. Then the attack is done by searching possible secret $\boldsymbol{s}_2' \in \mathcal{B}_{n,\leqslant h'}$ and check whether $\langle \boldsymbol{y}_1, \boldsymbol{b} \rangle - y_1^t A_2 \cdot \boldsymbol{s}_2'$ is far less than q or not. If there is such $\boldsymbol{s}_2'$ then we decide that the given sample is from LWE.

In this strategy, the probability that assumption holds is highly increased whereas the time complexity is not much increased; in practice the adversary choose $h' \lessapprox 10$ so that the dominated part is the lattice reduction algorithm. Thus this relaxation induces the smaller estimated security of LWE. We remark that this approach can be viewed as a tradeoff between dimension and error, as also noted by Albrecht.

Amortized Costs for Lattice Reductions.

To verify the guessed $\boldsymbol{s}_2'$ is correct or not, we should obtain several short $(\boldsymbol{y}_1, \boldsymbol{y}_2) \in \Lambda_{q,c}(A_1)$. To obtain several short vectors of similar length in a given lattice Λ, the easiest way would be repeating a lattice reduction that yields root Hermite factor δ_0, which gives vectors $\boldsymbol{v}_i$ of length less than $\delta_0^m \cdot \det \Lambda^{1/m}$.

Instead, Albrecht suggested a way that performs one expensive lattice reduction (e.g. BKZ_β) on given basis to have a sufficiently short ba-

sis B, and apply cheap lattice reductions (e.g. LLL) repeatedly while re-randomizing the short basis B by multiplying some short and sparse unimodular matrix U. Using sufficiently short and sparse U, the short vectors $\boldsymbol{v}_i$ obtained by this cheap lattice reduction which is estimated by

$$E(\|\boldsymbol{v}_i\|) = 2 \cdot \delta_0^m \cdot \det \Lambda^{1/m}.$$

For more details we refer [Alb17, Section 3].

To obtain statistically independent $(\boldsymbol{y}_1, \boldsymbol{y}_2) \in \Lambda_{q,c}(A_1)$, we have to assume that we can obtain arbitrarily many samples of DLWE. On the other hand, in many actual uses of LWE problem, there are only bounded number of samples $(A, \boldsymbol{b})$ are given; typically the number of samples would be $m = O(n)$. In this case we instead sample several short vectors $\boldsymbol{y}_i = (\boldsymbol{y}_{i,1} || \boldsymbol{y}_{i,2})$ in a fixed lattice $\Lambda_{q,c}(A_1)$. One can perform BKZ algorithm iteratively with re-randomizing basis, or can perform LLL algorithm iteratively according to the amortizing technique.

- Iterating BKZ: For a basis B of $\Lambda_{q,c}(A_1)$, iteratively perform BKZ on $B \cdot U$ while randomly sample arbitrary unimodular U.

- Iterating LLL: Perform BKZ on B to have B_{BKZ}. Randomly sample a *small and sparse* unimodular U, and run LLL on $B_{BKZ} \cdot U$ to have a short vector. Repeat this while changing unimodular U.

However, if we use the same lattice $\Lambda_{q,c}(A_1)$, new k-dimensional samples are not independent to each other anymore, since $\boldsymbol{y}_i$ comes from the same lattice $\Lambda_{q,c}(A_1)$. Thus we heuristically assume that, the short vectors $\boldsymbol{y}_i \in \Lambda_c(A_1)$ are independent to each other, that is, we still obtain $\mathrm{LWE}_{k,q,\chi}$ samples from $\boldsymbol{y}_i$.

Assumption 5.1.2. *Each iterative call of BKZ (or LLL) algorithm for randomized basis of $\Lambda_{q,c}(A_1)$ gives an independent short vector $\boldsymbol{y}_i$.*

5.2 Meet-in-the-Middle Attack on LWE

In this section, we describe an attack algorithm to solve LWE by meet-in-the-middle strategy. Let $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$ be $\mathrm{LWE}_{n,q,\alpha}(\mathcal{B}_{n,\leqslant h})$ samples with secret vector $\boldsymbol{s}$. For the MitM approach, it is natural to consider the noisy relation

$$A\boldsymbol{s}_1 \approx \boldsymbol{b} - A\boldsymbol{s}_2$$

for some $\boldsymbol{s}_1 \in \mathcal{B}_{n,\leqslant h/2}$ and $\boldsymbol{s}_2 \in \mathcal{B}_{n,\leqslant h/2}$ satisfying $\boldsymbol{s} = \boldsymbol{s}_1 + \boldsymbol{s}_2$. We first prepare a table

$$\mathcal{T} = \{A\boldsymbol{v}_1 \in \mathbb{Z}_q^m : \boldsymbol{v}_1 \in \mathcal{B}_{n,\leqslant h/2}\}^\dagger.$$

Then, we exhaustively investigate $\boldsymbol{v}_2 \in \mathcal{B}_{n,\leqslant h/2}$, while checking whether $\boldsymbol{b} - A\boldsymbol{v}_2 \in \mathbb{Z}_q^m$ is close to the set $\mathcal{T}$ where such closeness depends on the size of error $\boldsymbol{e}$. Now, if such case occurs for some $\boldsymbol{v}_2$, then we can expect that the vector $\boldsymbol{v}_2$ is the right half of secret $\boldsymbol{s}$. Otherwise, we cannot see such case for all possible $\boldsymbol{v}_2$, we conclude that the given sample is from the uniform distribution.

In this approach, finding an element in $\mathcal{T}$ that is close to $\boldsymbol{b}-A\boldsymbol{v}_2 \in \mathbb{Z}_q^m$ is the main task. A simple exhaustive method that checks every close vector to $\boldsymbol{b}-A\boldsymbol{v}_2 \in \mathbb{Z}_q^m$ surely works, but it costs too much time. We here resolve it by a search algorithm in the presence of noise that uses a locality sensitive

*Another way to use MitM method is to parse A and $\boldsymbol{s}$ into $[A_l|A_r]$ and $\boldsymbol{s} = (\boldsymbol{s}_l||\boldsymbol{s}_r)$ for $n/2$ dimension vectors. In the regards of the overall attack complexity that product of the time and the inverse of probability, the method discussed in the main body is better; The MitM with parsing takes less time and memory but the success probability is far less compared to the MitM in the paper.

hashing-like technique, which is adapted from Odlyzko's MitM attack on NTRU [HGSW03].

Before explaining our algorithm, we would like to remark that this MitM attack alone does not affect the practical parameter choice of the current schemes, but this attack serves as a main subroutine of our hybrid attack algorithm that will be introduced in Section 5.

Remark. To the best of our knowledge, there has been two papers that mentioned the MitM approach on LWE, but both of them are problematic; Bai and Galbraith [BG14] mentioned that there is a MitM attack on LWE, but they do not give the explicit algorithm, and Albrecht, Player and Scott [APS15] presented a MitM attack on LWE based on lexicographic order sorting, which has a flaw in the analysis. We describe this flaw in Appendix. We note that a very similar algorithm is considered in a different context; for example the inhomogeneous short integer solution problem under the name *approximate merge algorithm*.

5.2.1 Noisy Collision Search

For a vector $\boldsymbol{a} \in \mathbb{Z}_q^m$, we call a vector $\boldsymbol{t} \in \mathbb{Z}_q^m$ by B-noisy collision of $\boldsymbol{a}$ if $\|\boldsymbol{a} - \boldsymbol{t}\|_\infty \leqslant B$ for some $B < q/2$. Consider a set $\mathcal{T} \subset \mathbb{Z}_q^m$ and a vector $\boldsymbol{a} \in \mathbb{Z}_q^m$. Our purpose is to determine whether there is a B-noisy collision $\boldsymbol{t}$ of $\boldsymbol{a}$ in $\mathcal{S}$, and if so returns such vector $\boldsymbol{t}$. We mainly exploits a simple locality sensitive hashing $\mathsf{sgn} : \mathbb{Z}_q \to \{0, 1\}$, which defined as $\mathsf{sgn}(x) = 1$ for $x \in [0, q/2)$ and 0 otherwise. For every B-noisy collision $\boldsymbol{t} = (t_i)$ of $\boldsymbol{a} = (a_i)$, the sign of i-th entries $\mathsf{sgn}(a_i)$ and $\mathsf{sgn}(b_i)$ must coincide if $a_i \in V_B := [-q/2 + B, -B) \cup [B, q/2 - B)$.

For a vector $\boldsymbol{a} = (a_i) \in \mathbb{Z}_q^m$, define an index set $I_{\boldsymbol{a}} := \{i : a_i \in V_B\}$, and define a function $\mathsf{sgn}' : \mathbb{Z}_q \to \{0, 1, \mathsf{x}\}$ that returns $\mathsf{sgn}(a)$ if $a \in V_B$, and

otherwise x. Then from the above observation, we have the following fact that becomes a foundation of our algorithm

If $\mathcal{T}$ has a B-noisy collision of $\boldsymbol{a}$, then there is a binary string $(b_1, \cdots, b_m) \in \mathsf{sgn}(\mathcal{T})$ such that $b_i = \mathsf{sgn}'(a_i)$ for every index i in $I_{\boldsymbol{a}}$.

Detailed Algorithms.

We give two algorithms `Preprocess` and `Search`, where the former literally preprocess the set $\mathcal{T}$, and the latter investigate whether $\mathcal{T}$ has a B-noisy collision of input $\boldsymbol{a} \in \mathbb{Z}_q^m$.

- `Preprocess`: On input $\mathcal{T} \subset \mathbb{Z}_q^m$,

1. Initialize an empty hash table $\mathcal{H}$ with 2^m (empty) linked lists with indexes in $\{0,1\}^m$.
2. For each $\boldsymbol{t} \in \mathcal{T}$,
   (a) append $\boldsymbol{t}$ into the linked list indexed $\mathsf{sgn}(\boldsymbol{t})$.
3. Return nonempty linked lists $\mathcal{H}$.

- `Search`: On input a hash table $\mathcal{H}$, a query $\boldsymbol{a} \in \mathbb{Z}_q^m$ and distance bound B,

1. For each $\mathsf{bin} \in \{0,1\}^m$ obtained from $\mathsf{sgn}'(\boldsymbol{a})$ by replacing x by 0 or 1,
   (a) If $\mathcal{H}$ has a linked list indexed bin, for each $\boldsymbol{t}$ in the list,
      i. Check whether $\|\boldsymbol{a} - \boldsymbol{t}\|_\infty \leqslant B$. If so, return $\boldsymbol{t}$.
2. Return $\perp$.

Algorithm Analysis.

First, the following proposition asserts that our algorithm can find the B-noisy collision, if exists.

Proposition 5.2.1 (Correctness)**.** *Let $\mathcal{T}$ be a nonempty subset of $\mathbb{Z}_q^m$ and $\mathcal{H}$ be the output of Preprocess algorithm on input $\mathcal{T}$. Then* `Search` *algorithm with input $(\mathcal{L}, \boldsymbol{a}, B)$ returns a vector if and only if there is a B-noisy collision of $\boldsymbol{a}$ in $\mathcal{H}$. In particular, every returned vector is a B-noisy collision of $\boldsymbol{a}$.*

Proof. The second claim is immediate. For the first claim, one direction is clear since the output vector itself is a noisy collision in $\mathcal{T}$. Conversely, suppose that $\mathcal{T}$ has a noisy collision $\boldsymbol{t}$. Since $\mathsf{sgn}(\boldsymbol{t})$ would be one of strings obtained from $\mathsf{sgn}'(\boldsymbol{a})$, it outputs $\boldsymbol{t}$ unless it terminates before then with some vector $\boldsymbol{t}'$. □

To investigate the (time) cost of Algorithms, we presents some lemmas.

Lemma 5.2.1. *If $\boldsymbol{a} \xleftarrow{\$} \mathbb{Z}_q^m$, $|I_{\boldsymbol{a}}|$ follows a binomial distribution $B(m, 1 - 4B/q)$.*

Proof. Since $\boldsymbol{a}$ is sampled from $\mathcal{U}(\mathbb{Z}_q^m)$, the probability that each component a_i is not in V_B is $4B/q$. Each component of a_i is independent, and then we know the number of x in $\mathsf{sgn}'(\boldsymbol{a})$ follows a binomial distribution $B(m, 4B/q)$. □

Lemma 5.2.2. *Suppose the elements of the table $\mathcal{T}$ come from uniform distribution over $\mathbb{Z}_q^m$. For any $\mathsf{bin} \in \{0,1\}^m$,*

$$\Pr\left[L_{\mathsf{bin}} \neq \varnothing\right] \leqslant \frac{|\mathcal{T}|}{2^m}.$$

Proof. Note that $L_{\mathsf{bin}} \neq \varnothing$ if and only if $\mathsf{bin} \in \mathsf{sgn}(\mathcal{T})$. Since $\mathcal{T}$ is uniformly distributed, the probability of $\mathsf{bin} \notin \mathsf{sgn}(T)$ is $\left(1 - \frac{1}{2^m}\right)^{|\mathcal{T}|} \geqslant 1 - \frac{|\mathcal{T}|}{2^m}$, which proves the claim. □

Now assuming that the linked list insertion costs $O(1)$, the cost of `Preprocess` is clearly $O(|\mathcal{T}|)$. The costs of `Search` consists of $2^{m-|I_{\boldsymbol{a}}|}$ times of hash table lookups, and some computations of $\|\cdot\|_\infty$ norm. We first claim that $|I_{\boldsymbol{a}}|$ would be $m(1 - 4B/q)$ (stated in Lemma 5.2.1), which implies `Search` look ups the hash table about $2^{4mB/q}$ times.

Next we claim that by Heuristic 5.2.1, if m is sufficiently large[‡], the computation of $\|\cdot\|_\infty$ almost never occur for a randomly chosen query $\boldsymbol{a} \in \mathbb{Z}_q^m$.

Assumption 5.2.1. *Let $m, q > 0$ be positive integers and $B \in (0, q/4)$, and consider $\mathcal{T} \subset \mathbb{Z}_q^m$ whose element is sampled from uniform distribution. Let $\mathcal{H}$ be output of `Preprocess` on input $\mathcal{T}$. If*

$$m \geqslant 2\log(|\mathcal{T}|)/(1 - 4B/q), \tag{5.1}$$

then for a random vector $\boldsymbol{a} \leftarrow \mathbb{Z}_q^m$, the probability that `Search` never computes $\|\cdot\|_\infty$ norm is $\geqslant 1 - 1/|\mathcal{T}|$.

We justify the heuristic as follows: Since $|I_{\boldsymbol{a}}| = m(1 - 4B/q)$ for random $\boldsymbol{a} \in \mathbb{Z}_q^m$ on average by Lemma 5.2.1, we heuristically assume that `Search` visits $2^{4mB/q}$ indexes. Since $\Pr[L_{\mathsf{bin}} \neq \varnothing] \leqslant \frac{|\mathcal{T}|}{2^m}$ by Lemma 5.2.2, we bound the probability that `Search` never visits nonempty linked lists by $\left(1 - \frac{|\mathcal{T}|}{2^m}\right)^{4mB/q}$. One can easily check that if such choice of m yields the claim.

[‡]Note that, when we use noisy collision search to solve LWE, the parameter m is the number of samples of given LWE instances so it can be freely chosen by adversary.

`Preprocess`	`Search`[§]
$\lvert\mathcal{T}\rvert \cdot m$	$O(2^{4mB/q})$
(operations on $\mathbb{Z}_q$)	(table look-ups)

Table 5.1: Time cost for noisy search

Considering all above, we assess the total time cost in Table 5.1.

5.2.2 Noisy Meet-in-the-middle Attack on LWE

We now present a (noisy) MitM attack for LWE, using noisy collision search. Formal description is given by Algorithm 5. We would like to remark that, since we mainly exploit this algorithm as a subroutine of the main hybrid attack for LWE, Algorithm 5 is also described for LWE although it can actually solve the search version of LWE. Here, we define $\mathcal{B}_{n,h}$ by a set of vectors in $\{0, \pm 1\}_q^n$ with h number of nonzero entries. Also, $\mathcal{B}_{n,\leqslant h}$ denotes $\cup_{i=0}^{h} \mathcal{B}_{n,i}$.

One can easily check that correctness of Algorithm 5 comes immediately from the correctness of noisy search.

Proposition 5.2.2. *Let $h_1, h_2 > 0$ be positive integers, χ be a (B, ε)-bounded distribution over $\mathbb{Z}$, and let $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m \times (n+1)}$ be $\mathcal{A}_{n,q,\chi,s}^{LWE}$ samples where $\boldsymbol{s} \in \mathcal{B}_{n,\leqslant h_1+h_2}$. Then Algorithm 5 returns 1 for input $(A, \boldsymbol{b})$ and h_1, h_2 with probability $\geqslant (1-\varepsilon)^m$.*

Proof. If input $(A, \boldsymbol{b})$ is LWE sample with sparse ternary secret $\boldsymbol{s} \in \mathcal{B}_{n,\leqslant h_1+h_2}$, we exhaustively run the noise search on $\boldsymbol{v}_1 \in \mathcal{B}_{n,t_1}$ for $t_1 \leqslant h_1$ and $\boldsymbol{v}_2 \in \mathcal{B}_{n,t_2}$ for $t_2 \leqslant h_1$. These search should find $(\boldsymbol{s}_1, \boldsymbol{s}_2)$ such that

[§]Per one query in average.

Algorithm 5: Meet-in-the-middle attack for binary sparse LWE problems

Input : A matrix $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$
Hamming weight parameters $h_1, h_2 > 0$
Output: 1 if $(A, \boldsymbol{b})$ is from LWE distribution, and 0 otherwise

```
Compute T = {Av_1 : v_1 ∈ B_{n,⩽h_1}};
Run Preprocess on input T to have a hash table H;
for v = b − Av_2 ∈ Z_q^m for each v_2 ∈ B_{n,h_2} do
    if Search on input (H, v, B) returns a vector, then
        return 1
    end
end
return 0
```

$\boldsymbol{s} = \boldsymbol{s}_1 + \boldsymbol{s}_2$ and in this case the following equations holds:

$$\|A\boldsymbol{s}_1 - (\boldsymbol{b} - A\boldsymbol{s}_2)\|_\infty = \|A\boldsymbol{s} - \boldsymbol{b}\|_\infty = \|\boldsymbol{e}\|_\infty$$

Since Algorithm 5 returns 1 if $\|\boldsymbol{e}\|_\infty \leqslant B$ and each coordinate of error $\boldsymbol{e}$ follows χ, we conclude the algorithm succeeds with probability $\geqslant (1 - \varepsilon)^m$. □

To apply the analyses of noisy collision search, we need the following assumption that says that the vectors in table and queries are randomly distributed over $\mathbb{Z}_q^m$.

Assumption 5.2.2. *For a fixed matrix $A \in \mathbb{Z}_q^{m\times n}$, a distribution of vectors of the form $A\boldsymbol{s}$ where $\boldsymbol{s} \leftarrow \mathcal{B}_{n,\leqslant h}$ is sufficiently close to the uniform distribution over $\mathbb{Z}_q^m$.*

Proposition 5.2.3. *Suppose that Assumption 5.2.2 holds. Then for a uniformly random matrix $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$, Algorithm 5 returns 0 for input*

$(A, \boldsymbol{b})$ and parameters h_1, h_2 with probability $\geqslant 1 - N_{\mathcal{T}} N_q \cdot \left(\frac{2B}{q}\right)^m$, where $N_{\mathcal{T}}(n, h_1)$ and $N_q(n, h_2)$ denotes the number of vectors in table and the number of query.

Proof. By Assumption 5.2.2, we consider every query $\boldsymbol{v} = \boldsymbol{b} - A\boldsymbol{v}_2$ as a random sample from $\mathbb{Z}_q^m$. Then again from the assumption, the set $\mathcal{T}$ is randomly distributed on $\mathbb{Z}_q^m$, and we conclude that the probability that a B-noisy collision of $\boldsymbol{v}$ is in $\mathcal{T}$ is less than $N_{\mathcal{T}}(2B/q)^m$. Since we try at most N_q queries, the claim holds. □

Clearly, the time complexity of Algorithm 5 is the sum of table construction and `Preprocess` time T_{pre}, and total noisy search time T_{search}. Clearly, the size of table $N_{\mathcal{T}}$ and the number of query N_q is given by

$$N_{\mathcal{T}} = \sum_{i=1}^{h_1} \binom{n}{i} \cdot 2^i, \;\; N_q = \sum_{i=1}^{h_2} \binom{n}{i} \cdot 2^i \tag{5.2}$$

for given h_1, h_2. Finally, by supposing Assumption 5.2.2 holds and the condition for m (5.1), we have the following cost estimation.

- T_{pre} consists of $N_{\mathcal{T}} \cdot n^2$ operations over $\mathbb{Z}_q$ on constructing table $\mathcal{T}$, and `Preprocess` also requires $N_{\mathcal{T}} \cdot m$ operations.
- Since each `Search` call for each query costs $2^{4mB/q}$ in average, we have $T_{search} = O(N_q \cdot 2^{4mB/q})$.

5.3 The Hybrid-Dual Attack

In this section, we propose a hybrid attack that combines lattice reduction and the MitM attack. More precisely, we use dual attack as a trade-off

Memory	Time	
	T_{pre}	T_{search}
$N_{\mathcal{T}} \cdot m$ (bits)	$N_{\mathcal{T}} \cdot (n^2 + m)$ (operations)	$O(N_q \cdot 2^{4mB/q})$ (table look-ups)

Table 5.2: Cost for Algorithm 5 with inputs a matrix in $\mathbb{Z}_q^{m\times(n+1)}$ and h_1, h_2.

method for LWE sample, which increases the error size and reduces dimension and Hamming weight of secret vector. For that MitM attack of the previous section cost heavily depends on the dimension of secret vector but less sensitive to error size, this trade-off largely decreases the MitM attack cost.

5.3.1 Dimension-error Trade-off of LWE

In this section we interpret Albrecht's dual attack as dimension-error trade-off with detailed analysis. For given LWE samples $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$ from $\mathcal{A}_{n,q,\alpha,\boldsymbol{s}}^{LWE}$ for $k < n$, divide A into A_1 and A_2 consisting of the first $n-k$ columns and the remaining k columns. For any vectors $(\boldsymbol{y}_1, \boldsymbol{y}_2) \in \Lambda_{q,c}(A_1)$, it holds that

$$\begin{aligned}\langle \boldsymbol{y}_1, \boldsymbol{b}\rangle &= \langle \boldsymbol{y}_1, A_1\boldsymbol{s}_1\rangle + \langle \boldsymbol{y}_1, A_2\boldsymbol{s}_2\rangle + \langle \boldsymbol{y}_1, \boldsymbol{e}\rangle \\ &\equiv_q \boldsymbol{y}_1^t A_2\boldsymbol{s}_2 + c\cdot \boldsymbol{y}_2^t\boldsymbol{s}_1 + \boldsymbol{y}_1^t\boldsymbol{e}\end{aligned}$$

where $\boldsymbol{s}_2$ is the last k entries of $\boldsymbol{s}$. Now, if $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ is sufficiently short to satisfy $\langle \boldsymbol{y}_1, \boldsymbol{e}\rangle, \langle \boldsymbol{y}_2, \boldsymbol{s}_1\rangle \ll q$, we have a new LWE-like sample

$$(\boldsymbol{y_1}^t A_2, \langle \boldsymbol{y}_1, \boldsymbol{b}\rangle) = (\boldsymbol{a}', \langle \boldsymbol{a}', \boldsymbol{s}_2\rangle + e') \in \mathbb{Z}_q^{k+1},$$

with new secret vector $\boldsymbol{s}_2$ and error $e' = c\cdot\langle \boldsymbol{y}_2, \boldsymbol{s}_1\rangle + \langle \boldsymbol{y}_1, \boldsymbol{e}\rangle$.

Algorithm 6: A Dimension-error Trade-off

Input : A matrix $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m \times (n+1)}$
Root Hermite factor δ_0
Dimension trade-off parameter $0 < k < n$
Output: A vector $(\boldsymbol{a}', b') \in \mathbb{Z}_q^{k+1}$.

1 Parse A into $[A_1||A_2]$ with $A_1 \in \mathbb{Z}_q^{m \times (n-k)}$ and $A_2 \in \mathbb{Z}_q^{m \times k}$;
2 $\boldsymbol{y} = (\boldsymbol{y}_1||\boldsymbol{y}_2) \leftarrow BKZ_{\delta_0}(\Lambda_{q,c}^{\perp}(A_1))$;
3 **return** $(\boldsymbol{a}', b') \leftarrow (\boldsymbol{y}_1^t A_2, \langle \boldsymbol{y}, \boldsymbol{b} \rangle) \in \mathbb{Z}_q^{k+1}$.

We now have Algorithm 6 for the dimension-error trade-off, while assuming Assumption 5.1.1 to justify the choice for c in Section 3.1. In other words, we choose $c = \frac{\alpha q}{\sqrt{2\pi}} \cdot \frac{\|\boldsymbol{y}_1\|}{|\langle \boldsymbol{y}_2, \boldsymbol{s}_1 \rangle|}$ and assume that each entry of $\boldsymbol{y}$ has similar size $\|\boldsymbol{y}\|/\sqrt{m+n}$. We formally state that Algorithm 6 can serve a trade-off algorithm on the LWE problem as follows.

Proposition 5.3.1. *Assume that Assumption 5.1.1 holds for outputs of BKZ algorithm with root-Hermite factor δ_0. Then for given $\mathcal{A}_{n,q,\alpha,\boldsymbol{s}}^{LWE}$ samples $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m \times n}$, Algorithm 6 returns one $\mathcal{A}_{k,q,\chi,\boldsymbol{s}'}^{LWE}$ sample $(\boldsymbol{a}', b') \in \mathbb{Z}_q^{k+1}$, where $\boldsymbol{s}' = (s_{n-k+1}, \cdots, s_n)$. In particular, the error distribution χ is $(B, 2e^{-4\pi})$-bounded with*

$$B = \left(2 + \frac{1}{\sqrt{2\pi}}\right) \cdot \sqrt{\frac{m}{m+n}} \cdot \alpha q \cdot \|\boldsymbol{y}\| \tag{5.3}$$

Proof. It only remains to show the error bound part, and we use the following lemma.

Lemma 5.3.1 (Lemma 2.4 of [Ban95])**.** *For any real $s > 0$ and $C > 0$,*

and any $\boldsymbol{x} \in \mathbb{R}^n$*, we have*

$$\Pr[|\langle \boldsymbol{x}, \mathcal{D}_{\mathbb{Z}^n,s} \rangle| \geqslant C \cdot s\|\boldsymbol{x}\|] < 2 \cdot \exp(-\pi \cdot C^2).$$

From this lemma we know $\|\langle \boldsymbol{y}_1, \boldsymbol{e} \rangle\| < 2\alpha q \cdot \|\boldsymbol{y}_1\|$ with probability $\geqslant 1 - 2e^{-4\pi}$. Therefore, with probability $\geqslant 1 - 2e^{-4\pi}$, we have

$$\begin{aligned} |e'| &\leqslant |\langle \boldsymbol{y}_1, \boldsymbol{e} \rangle| + |c \cdot \langle \boldsymbol{y}_2, \boldsymbol{s} \rangle| \\ &\leqslant 2\alpha q \cdot \|\boldsymbol{y}_1\| + \frac{\alpha q}{\sqrt{2\pi}} \cdot \|\boldsymbol{y}_1\| \\ &\leqslant (2 + \frac{1}{\sqrt{2\pi}}) \cdot \alpha q \cdot \|\boldsymbol{y}_1\|. \end{aligned}$$

Since Assumption 5.1.1 guarantees $\|\boldsymbol{y}_1\| \approx \sqrt{\frac{m}{m+n}}\|\boldsymbol{y}\|$, we show (5.3). □

Amortizing and Heuristic for Algorithm 6

We remark that Albrecht's amortizing technique and heuristic assumption described in Section 3 works well for this trade-off. More precisely, the amortizing technique reduces the time cost for multiple run of tradeoff algorithm into, essentially, the time cost of one run of Algorithm 6. On the other hand, we can obtain arbitrary many independent trade-offed LWE samples from the bounded number, e.g. $m = O(n)$, of given LWE samples under the heuristic assumption. We employ these techniques in the hybrid attack and estimation as well.

5.3.2 Our Hybrid Attack

Now we are able to describe our hybrid attack, which is formally written in Algorithm 7. We first explain how to choose parameters m and τ optimally

from inputs.

- The number of n-dim DLWE samples m is set to minimize the short vectors obtained from BKZ_{δ_0}, precisely

$$m = \sqrt{\frac{n \log q/c}{\log \delta_0}}.$$

- The error bound B is subsequently obtained from m by Proposition 5.3.1, precisely

$$B = (2 + \frac{1}{\sqrt{2\pi}}) \cdot \alpha q \sqrt{\frac{m}{m+n}} \cdot 2^{2\sqrt{n \log \delta_0 \log q/c}}.$$

- The number of k-dim DLWE samples τ is chosen according to Heuristic 5.2.1[¶], in order to ensure that Algorithm 5 runs in time proportional to $2^{4\tau B/q}$, precisely

$$\tau = \frac{1}{1 - 4B/q} \log(N_{\mathcal{T}} \cdot N_q),$$

where

$$N_{\mathcal{T}} = |\mathcal{T}| = \sum_{i=1}^{h_1} \binom{k}{i} \cdot 2^i, \;\; N_q = \sum_{i=1}^{h_2} \binom{k}{i} \cdot 2^i.$$

The following theorem shows the results of Algorithm 7 for LWE samples.

[¶]We note that the parameter τ does not critically affect to the performance when we use the amortization technique. Hence we choose τ as in heuristical computation.

Algorithm 7: A new hybrid attack for sparse ternary secret LWE

Input : $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$
Root Hermite factor δ_0
Dimension trade-off parameter $0 < k < n$
MitM parameter $0 \leqslant h_1, h_2 \leqslant h$

Output : 1 if $(\boldsymbol{a}_i, b_i)$'s are sampled from LWE distribution, and 0 otherwise.

Set m, B and τ as optimal values;
`// Dimension-error trade-off;`
for *i from* 1 *to* τ **do**
  Let $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$ be $\mathrm{DLWE}_{n,q,\alpha}(\mathcal{B}_{n,h})$ samples;
  Run Algorithm 6 on input $(A, \boldsymbol{b})$, δ_0, and k to obtain $(\boldsymbol{a}'_i, b'_i) \in \mathbb{Z}_q^{k+1}$
end
$(A', \boldsymbol{b}') \in \mathbb{Z}_q^{\tau\times(k+1)}$ be a matrix having i-th row $(\boldsymbol{a}'_i, b'_i)$;
`// No need to perform MitM if` $\boldsymbol{s}_2 = \boldsymbol{0}$;
if $\|\boldsymbol{b}'_i\|_\infty \leqslant B$ **then**
  return 1
end
`// Perform MitM;`
if *Algorithm 5 on input* $(A', \boldsymbol{b}'), B, h_1,$ *and* h_2 *outputs* 1 **then**
  return 1
end
return 0

Theorem 5.3.1. *Let* $\boldsymbol{s} \in \mathcal{B}_{n,h}$. *Given sufficiently many* $\mathcal{A}^{LWE}_{n,q,\alpha,\boldsymbol{s}}$ *samples, Algorithm 7 returns* 1 *with probability*

$$p = (1 - 2e^{-4\pi})^m \cdot \sum_{0\leqslant i\leqslant h_1+h_2} \binom{n-h}{k-i}\binom{h}{i} / \binom{n}{k}.$$

Proof. Let the secret vector $\boldsymbol{s}$ be $\boldsymbol{s} = (\boldsymbol{s}_1 \| \boldsymbol{s}_2)$ which is seperated as $\boldsymbol{y} = (\boldsymbol{y}_1 \| \boldsymbol{y}_2)$. This means that we run Algorithm 5 by input $(A', \boldsymbol{b}')$, which has $\boldsymbol{s}_2$ as its LWE secret. Thus Algorithm 5 returns 1 if and only if $\mathsf{HW}(\boldsymbol{s}_2) \leqslant$

Memory	Time		
	T_{lat}	T_{pre}	T_{search}
$N_{\mathcal{T}} \cdot \tau$ (bits)	$\approx T_{BKZ,\delta_0}$	$N_{\mathcal{T}} \cdot (k^2 + \tau)$ (operations)	$O(N_q \cdot 2^{4\tau B/q})$ (table look-ups)

Table 5.3: Cost for Algorithm 7

$h_1 + h_2$. This probability is

$$p' = \sum_{0 \leqslant i \leqslant h_1 + h_2} \binom{n-h}{k-i} \binom{h}{i} / \binom{n}{k}.$$

From the choice of B and Proposition 5.3.1, we get $p = (1 - 2e^{-4\pi})^m \cdot p'$. □

Under the amortizing technique and heuristic assumption, the time cost of the trade-off phase is approximately one lattice reduction, and the condition *sufficiently many* is removed. Overall, the total time complexity of Algorithm 7 is dominated by the sum of lattice reduction time T_{lat} and Algorithm 5 time $T_{pre} + T_{search}$. Since we take τ according to Heuristic 5.2.1, the table 5.2 is also applicable to this case, which yields the following time cost table with the amortizing technique.

5.4 The Hybrid-Primal Attack

In this chapter, we revisit the hybrid attack in the context of the LWE problem using sparse and ternary secret, together with various techniques derived from other LWE attack literature. To distinguish from the hybrid-dual attack of the previous section, we call this attack by the hybrid-primal attack. We further refine the analysis of the hybrid attack to be align with LWE setting, and derive more accurate and reliable security estimate.

Upon our analysis, we estimate the complexity of the hybrid attack

for various parameters currently used in the HE literature. As a result, we observe that the hybrid attack outperforms the previously considered attacks[‖] on currently used HE parameter regime which urges parameter update to maintain the same security level.

We finally remark that, our result re-ensures that the security implication of the use of sparse secret is not well understood yet, as the homomorphic encryption standardization states.

5.4.1 The Primal Attack on LWE

The primal lattice attack for LWE solves the bounded distance decoding (BDD) problem directly. That is, given LWE samples $(A, \boldsymbol{b})$, it finds a vector $\boldsymbol{w} = A\boldsymbol{s}$ such that $\|\boldsymbol{b}-\boldsymbol{w}\|$ is unusually small. The literature has mainly considered two approaches to solve BDD: the first one directly solves BDD using Babai's nearest algorithm followed by lattice reduction [LP11], and the second one converts the BDD instance into (u)SVP instance, and solves it by lattice reduction [ADPS16, AGVW17]. We here only explain the second method that is more widely considered. For this method one converts the given LWE samples into some lattice. The *Kannan* embedding [Kan87] considers the column echelon form $[I_n||A'^t]^t$ of $A \in \mathbb{Z}_q^{m\times n}$ (after appropriate permutation of rows) and construct the lattice Λ_{Kan} generated by the following matrix

$$B_{Kan} = \begin{pmatrix} qI_{m-n} & A' & \\ 0 & I_n & \boldsymbol{b} \\ 0 & 0 & 1 \end{pmatrix} \in \mathbb{Z}^{(m+1)\times(m+1)}$$

[‖]For the previous attack estimation, we exploit LWE-estimator [APS15].

which has a short vector $(\boldsymbol{e}, 1) \in \mathbb{Z}^{m+1}$. However, this approach cannot benefit when the secret is small, which information may lead to better attack by allowing the attacker to exploit it.

In this regard, another lattice embedding is proposed by [BG14]:

$$\Lambda_{BG} = \{\boldsymbol{x} \in \mathbb{Z}^m \times (\nu\mathbb{Z})^n \times \mathbb{Z}\} : \left(I_m \mid\mid \frac{1}{\nu}A \mid\mid -\boldsymbol{b}\right) \cdot \boldsymbol{x} = \boldsymbol{0} \bmod q\}.$$

This lattice contains an unusual short vector $(\boldsymbol{e}, \nu\boldsymbol{s}, 1)$. Thus, we can find the secret vector $\boldsymbol{s}$ along with error vector $\boldsymbol{e}$ by solving SVP on a lattice generated by basis

$$B_{BG,\nu} = \begin{pmatrix} qI_m & A & -\boldsymbol{b} \\ 0 & \nu I_n & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The scaling factor ν is determined so that the short vector $(\boldsymbol{e}, \nu\boldsymbol{s}, 1)$ is balanced, or explicitly

$$\|\boldsymbol{e}\| \approx \|\nu\boldsymbol{s}\|.$$

Upon the choice of such ν, the vector $(\boldsymbol{e}, \nu\boldsymbol{s}, 1)$ is assumed to be of the form $(\boldsymbol{e}', 1)$ where $\boldsymbol{e}'$ is sampled from Gaussian distribution having same standard deviation with $\boldsymbol{e}$.

Unique-SVP estimate

One attack model based on the primal strategy was proposed in [ADPS16] and rigorously analyzed in [AGVW17]. We remark that, the `usvp` tab of `LWE-estimator` currently considers this attack model. When the BKZ algorithm is applied for a random d-dimensional lattice, the SVP oracle finds the shortest vector of the last projected lattice of size β, whose length is

expected to be

$$\delta_0^{2\beta-d} \cdot \det(\Lambda(B))^{1/d}$$

under GSA assumption. Meanwhile, in the embedding lattice for the primal strategy, the projection of $(\boldsymbol{e}, 1)$ to the last β Gram-Schmidt vectors has size

$$\sqrt{\beta/d} \cdot \|(\boldsymbol{e}, 1)\| \approx \sqrt{\beta}\sigma$$

where σ is the standard deviation of each component of $\boldsymbol{e}$. Upon this facts, [AGVW17] argues and confirms on an experimental basis that, for β satisfying

$$\sqrt{\beta}\sigma \leqslant \delta_0^{2\beta-d} \cdot \det(\Lambda(B))^{1/d}, \tag{5.4}$$

one can totally recover the short vector using BKZ with such β.

Sparse secret case

When the secret is further assumed to be sparse, most of columns of A are irrelevant to $\boldsymbol{b} = A\boldsymbol{s} + \boldsymbol{e}$. From this observation, one can randomly remove some columns of $A \in \mathbb{Z}_q^{m\times n}$ to have $A' \in \mathbb{Z}_q^{k\times n}$ $(k < n)$, and then apply the primal strategy to $(A', \boldsymbol{b})$ that requires smaller blocksize β for (5.4). This succeeds if $(A', \boldsymbol{b})$ is also LWE samples, or equivalently, all the removed columns correspond to zero component of $\boldsymbol{s}$. Note that it happens with adequate probability, say p_k, due to sparsity of the secret. Considering this into account, the attack complexity for sparse secret is calculated by

$$\min_k \frac{1}{p_k} \cdot T_k$$

where T_k is the time cost for the primal attack on k-dimensional LWE sample.

5.4.2 The Hybrid Attack for SVP

In this section we recall the description and bird-eye analysis flow of the hybrid attack [Wun16]. Generally, the hybrid attack finds a short vector $\boldsymbol{v} = (\boldsymbol{v}_l, \boldsymbol{v}_g)$ in a lattice Λ, whose basis is of the form

$$B = \begin{pmatrix} T & C \\ 0 & I_r \end{pmatrix} \in \mathbb{Z}^{(d+r)\times(d+r)}.$$

For our interest case, we assume that $\boldsymbol{v}_l$ is sampled from a small Gaussian distribution $\mathcal{D}^d_{\alpha q}$ and $\boldsymbol{v}_g$ is ternary vector having low Hamming weight $h \leqslant r$.

Hybrid with Exhaustive-search

The main observation for the hybrid attack is

$$\boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}_l \\ \boldsymbol{v}_g \end{pmatrix} = B \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{v}_g \end{pmatrix} = \begin{pmatrix} T\boldsymbol{x} + C\boldsymbol{v}_g \\ \boldsymbol{v}_g \end{pmatrix}$$

for some $\boldsymbol{x}$. Then we have $\boldsymbol{v}_l = T\boldsymbol{x} + C\boldsymbol{v}_r$, which implies

$$\mathsf{NP}_T(C\boldsymbol{v}_g) = \mathsf{NP}_T(T\boldsymbol{x} + C\boldsymbol{v}_g) = \mathsf{NP}_T(\boldsymbol{v}_l).$$

From this we consider the following hybrid attack of lattice reduction and exhaustive search:

1. Reduce the matrix T so that $\mathsf{NP}_T(\boldsymbol{v}_l) = \boldsymbol{v}_l$
2. Guess $\boldsymbol{v}_r$ and compute $\mathsf{NP}_T(C\boldsymbol{v}_r)$; if the guess is correct, one has unusually short result, namely $\boldsymbol{v}_l$.

The detailed procedure is given below by Algorithm 8.

Algorithm 8: A Hybrid of Exhaustive Search

Input : A matrix $B = \begin{pmatrix} T & C \\ 0 & I_r \end{pmatrix} \in \mathbb{Z}^{(d+r)\times(d+r)}$
A blocksize β
A weight parameter h_g
An expected bound y for $\|\boldsymbol{v}_l\|_\infty$.
Output : A short vector $\boldsymbol{v}$ in $\Lambda(B)$

$T \leftarrow \mathsf{BKZ}_\beta(T)$;
for *for each* $\boldsymbol{w} \in \{\pm 1, 0\}^r$ *of Hamming weight* h_g **do**
  $\boldsymbol{v}'_l \leftarrow \mathsf{NP}_T(C\boldsymbol{w}) \in \mathbb{Z}^d$;
  if $\boldsymbol{v} = (\boldsymbol{v}'_l || \boldsymbol{v}_g) \in \Lambda(B)$ *and* $\|\boldsymbol{v}_l\|_\infty \leqslant y$ **then**
    return $\boldsymbol{v}$.
  end
end
return `False`

Speedup with MitM

Upon this basic attack, one can speed up the guessing step by MitM approach. For two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ of low weight satisfying $\boldsymbol{v}_g = \boldsymbol{v}_1 + \boldsymbol{v}_2$, we have

$$C\boldsymbol{v}_1 = -C\boldsymbol{v}_2 + C\boldsymbol{v}_g = -C\boldsymbol{v}_2 + \boldsymbol{v}_l - T\boldsymbol{x},$$

and hence

$$\mathsf{NP}_T(C\boldsymbol{v}_1) = \mathsf{NP}_T(-C\boldsymbol{v}_2 + \boldsymbol{v}_l).$$

For MitM strategy, one hopes that the NP algorithm works homomorphically, that is,

$$\mathsf{NP}_T(-C\boldsymbol{v}_2 + \boldsymbol{v}_l) = \mathsf{NP}_T(-C\boldsymbol{v}_2) + \mathsf{NP}_T(\boldsymbol{v}_l) \tag{5.5}$$

in order to have

$$\mathsf{NP}_T(C\boldsymbol{v}_1) = \mathsf{NP}_T(-C\boldsymbol{v}_2 + \boldsymbol{v}_l) = \mathsf{NP}_T(-C\boldsymbol{v}_2) + \mathsf{NP}_T(\boldsymbol{v}_l).$$

As we reduce the matrix T so that $\mathsf{NP}_T(\boldsymbol{v}_l) = \boldsymbol{v}_l$, one reaches

$$\mathsf{NP}_T(C\boldsymbol{v}_1) = \mathsf{NP}_T(-C\boldsymbol{v}_2) + \boldsymbol{v}_l \approx \mathsf{NP}_T(-C\boldsymbol{v}_2) = -\mathsf{NP}_T(C\boldsymbol{v}_2) \tag{5.6}$$

from which one tries to detect the (noisy) collision in MitM manner. The event (5.5) definitely not always happens, and indeed the probability for (5.5) plays a crucial role to analyze the attack complexity.

To detect the collision, we need to store vector $\boldsymbol{v}$ in a table having addresses related to $\mathsf{NP}(C\boldsymbol{v})$. In this regard, we define the address set $\mathcal{A}_{\boldsymbol{x}}$ below: note that for a bound y such that $\|\boldsymbol{v}_l\|_\infty \leqslant y$, we have

$$\mathcal{A}^{(d,y)}_{\mathsf{NP}_T(C\boldsymbol{v}_1)} \cap \mathcal{A}^{(d,y)}_{-\mathsf{NP}_T(C\boldsymbol{v}_2)} \neq \varnothing,$$

which enables one to find the collision.

Definition 5.4.1 (Definition 1 of [Wun16])**.** *For a vector $\boldsymbol{x} \in \mathbb{Z}^d$ the set $\mathcal{A}^{(d,y)}_{\boldsymbol{x}} \subset \{0,1\}^d$ is defined as*

$$\mathcal{A}^{(d,y)}_{\boldsymbol{x}} = \left\{ \boldsymbol{a} \in \{0,1\}^d : \begin{array}{llll} a_i = 1 & if & x_i > \lceil \frac{y}{2} - 1 \rceil \\ a_i = 0 & if & x_i < \lfloor -\frac{y}{2} \rfloor \end{array} \right\}.$$

Algorithm 9 below describes the detail. The main loop investigates vectors of Hamming weight h_M, while expecting $\boldsymbol{v}_g$ is represented by the sum of two vectors of weight h_M. Note that this happens not only for $\mathsf{HW}(\boldsymbol{v}_g) = 2h_M$ case, but $\mathsf{HW}(\boldsymbol{v}_g) = 2k$ for some $k \leqslant h_M$ case.

Algorithm 9: A Hybrid MitM Attack

Input : A matrix $B = \begin{pmatrix} T & C \\ 0 & I_r \end{pmatrix} \in \mathbb{Z}^{(d+r)\times(d+r)}$
A blocksize β
A weight parameter h_M
An expected bound y for $\|\boldsymbol{v}_l\|_\infty$.
Output : A short vector $\boldsymbol{v}$ in $\Lambda(B)$

$T \leftarrow \mathsf{BKZ}_\beta(T)$;
for *each* $\boldsymbol{w} \in \{\pm 1, 0\}^r$ *of Hamming weight* h_M **do**
  $\boldsymbol{v}'_l \leftarrow \mathsf{NP}_T(C\boldsymbol{w}) \in \mathbb{Z}^d$;
  store $\boldsymbol{w}$ in all the boxes having address in a set $\mathcal{A}^{(d,y)}_{\boldsymbol{v}'_l} \cup \mathcal{A}^{(d,y)}_{-\boldsymbol{v}'_l}$;
  for *each* $\boldsymbol{w}' \neq \boldsymbol{w}$ *in all boxes of address in* $\mathcal{A}^{(d,y)}_{\boldsymbol{v}'_l} \cup \mathcal{A}^{(d,y)}_{-\boldsymbol{v}'_l}$ **do**
    $\boldsymbol{v}_g \leftarrow \boldsymbol{w} + \boldsymbol{w}'$ and $\boldsymbol{v}_l \leftarrow \mathsf{NP}_T(C\boldsymbol{v}_g) \in \mathbb{Z}^{d-r}$;
    if $\boldsymbol{v} = (\boldsymbol{v}_l || \boldsymbol{v}_g) \in \Lambda(B)$ *and* $\|\boldsymbol{v}_l\|_\infty \leqslant y$ **then**
      return $\boldsymbol{v}$.
    end
  end
end
return `False`

Analysis for MitM hybrid

The time cost of Algorithm 9 and its main parts consist of the lattice reduction cost T_{BKZ} and the guessing cost T_{guess}. The reduction cost T_{BKZ} can be easily estimated from blocksize β and dimension $d - r$, and hence in the following we mainly focus on T_{guess}.

We estimate T_{guess} by one inner loop cost multiplied by the expected number of loops, say L, and for the sake of simplicity, we establish the following assumption.

Assumption 5.4.1. *We assume that one inner loop cost of Algorithm 9 is dominated by nearest plane algorithm cost T_{NP}.*

Explanation. This assumption is closely related to the expected bound y of $\|\boldsymbol{v}_l\|_\infty$: Too small y makes the algorithm fail to find the answer, and too large y increases the size of address set so that Assumption 5.4.1 fails. We will consider

$$y = 6\frac{\alpha q}{\sqrt{2\pi}},$$

that is 6 times of standard deviation of $\mathcal{D}_{\alpha q}$. Indeed, this value is sufficiently large so that $\|\boldsymbol{v}_l\|_\infty \leqslant y$ holds with high probability, and sufficiently small so that Assumption 5.4.1 makes sense. To be precise, we justify the followings for our interest parameters.

- We have
$$\Pr_{\boldsymbol{v}_l \leftarrow \mathcal{D}_{\alpha q}^d}[\|\boldsymbol{v}_l\|_\infty \leqslant y] \geqslant 0.99.$$
- For $\boldsymbol{x} \leftarrow \mathcal{P}(T^*)$, the address set $\mathcal{A}_{\boldsymbol{x}}^{(d,y)}$ consists of only one element with overwhelming probability.

For the first claim, note that the probability $\Pr_{e\leftarrow\mathcal{D}_{\alpha q}}[|e| \geqslant y]$ is about 2^{-28} by approximating the discrete Gaussian as a continuous one. Then $\|\boldsymbol{v}_l\| \leqslant y$ with probability at least $(1-d\cdot 2^{-28})$, and since our all parameters satisfy $d \leqslant 2^{20}$, this is still larger than 0.99.

We now explain the second claim. From the definition, one can check that the number of address set $\mathcal{A}_{\boldsymbol{x}}^{(d,y)}$ is 2^{ℓ} where ℓ is the number of components of $\boldsymbol{x}$ in $[-\frac{y}{2}, \frac{y}{2}]$. Then for a random choice of $\boldsymbol{x} \leftarrow \mathcal{P}(T^*)$, the probability of x_i is in $[-\frac{y}{2}, \frac{y}{2}]$ is $\frac{y}{R_i}$ where R_i is the i-th Gram-Schmidt length of T. Then we establish an expectation for ℓ by

$$E[\ell] = \sum_{i=1}^{d} \frac{y}{R_i}.$$

By assuming GSA, we have an upper bound for that expectation by

$$\ell \leqslant d \cdot \frac{y}{R_d} = d \cdot \frac{y}{\delta_0^{-d} \cdot \det(T)^{1/d}}.$$

For all of our parameters in Table 5.4 one can check that the right hand side value is much smaller than 1. □

From Assumption 5.4.1, we have $T_{guess} = L \cdot T_{\mathsf{NP}}$ where $T_{\mathsf{NP}} = d^2/2^{1.06}$ according to (2.2). Toward an estimation for L, we start by defining two sets

$$W = \{\boldsymbol{w} \in \{\pm 1, 0\}^r : \mathsf{HW}(\boldsymbol{w}) = h_M\}$$

and

$$V = \{\boldsymbol{w} \in W : (\boldsymbol{v}_g - \boldsymbol{w} \in W) \wedge (\mathsf{NP}_T(C\boldsymbol{w}) + \mathsf{NP}(C\boldsymbol{v}_g - C\boldsymbol{w}) = \boldsymbol{v}_l)\},$$

and two probabilities

$$p_s := \Pr_{\substack{\boldsymbol{w}\leftarrow W \\ \boldsymbol{v}_l \leftarrow \mathcal{D}^d_{\alpha q}}} [\mathsf{NP}_T(C\boldsymbol{w}) + \mathsf{NP}_T(C\boldsymbol{v}_g - C\boldsymbol{w}) = \boldsymbol{v}_l]$$

and

$$p_c := \Pr_{\boldsymbol{w}\leftarrow W}[\boldsymbol{v}_g - \boldsymbol{w} \in W]$$

for which we make the following assumption.

Assumption 5.4.2. *We assume that two probabilities p_s and p_c are independent, and further assume that*

$$|V| = p_s p_c |W|.$$

Explanation. We will apply this analysis for the MitM speed-up only when

$$|W| \geqslant \frac{1}{p_s p_c}.$$

If this inequality is unsatisfied with given parameters, the set V is likely to be empty and Lemma 5.4.1 becomes vacuous, and hence this analysis for the MitM speed-up becomes utterly improper. □

Regarding the set V, the following lemma gives an algorithm terminates condition.

Lemma 5.4.1. *Algorithm 9 terminates with $\boldsymbol{v}_g$ right after the main loop chooses two vectors $\boldsymbol{v}_1, \boldsymbol{v}_2 \in V$ such that $\boldsymbol{v}_1 + \boldsymbol{v}_2 = \boldsymbol{v}_g$.*

Proof. Since $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ belong to V, we have $\mathsf{NP}_T(C\boldsymbol{v}_1) + \mathsf{NP}(C\boldsymbol{v}_2) = \boldsymbol{v}_l$. Then $\mathsf{NP}_T(C\boldsymbol{v}_1)$ and $-\mathsf{NP}(C\boldsymbol{v}_2)$ differ by $\boldsymbol{v}_l$, and hence from the definition of address set, we have $\mathcal{A}_{\mathsf{NP}(C\boldsymbol{v}_1)} \cap \mathcal{A}_{-\mathsf{NP}(C\boldsymbol{v}_2)} \neq \varnothing$. Thus $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are

stored in at least one box, and Algorithm 9 detects them and return $\boldsymbol{v}_g = \boldsymbol{v}_1 + \boldsymbol{v}_2$. □

From Assumption 5.4.2, we expect that the main loop samples one vector in V for every $\frac{1}{p_s p_c}$ repeats, and by Lemma 5.4.1 we estimate the number of loops are estimated by the birthday paradox as

$$L \approx \frac{\sqrt{|V|}}{p_s p_c} = \sqrt{\frac{|W|}{p_s p_c}} = \sqrt{\frac{2^{h_M}\binom{r}{h_M}}{p_s p_c}}. \tag{5.7}$$

It remains to compute the probabilities p_s and p_c to completely represent (5.7) by the parameters d, β, r and h_M. Rather than giving too generalized formula for this, we postpone this later in Section 5.4.4 after we give the detail for the hybrid attack against LWE case.

5.4.3 The Hybrid-Primal attack for LWE

In this section, we apply the hybrid attack algorithm to the primal lattice attack against LWE, and adapt previous analysis in accordance with our interest LWE setting: small and sparse secret with (discrete) Gaussian error. Without any special mention, we assume that LWE sample $(A, \boldsymbol{b})$ is given by $\mathsf{LWE}_{n,q,\alpha}(\mathcal{B}_h)$.

Overview

Given LWE sample $(A, \boldsymbol{b} = A\boldsymbol{s} + \boldsymbol{e})$, we consider Bai-Gal embedding with some change of the order columns and $\nu = 1$

$$B = \begin{pmatrix} qI_m & -\boldsymbol{b} & A \\ 0 & 1 & 0 \\ 0 & 0 & I_n \end{pmatrix}$$

that contains a short vector $\boldsymbol{v} = (\boldsymbol{e}, 1, \boldsymbol{s})$. By taking a MitM dimension parameter $r \leqslant n$, we divide the matrix by following:

$$B = \left(\begin{array}{cc|c} qI_m & * & * \\ 0 & I_{n+1-r} & 0 \\ \hline 0 & 0 & I_r \end{array}\right),$$

and parse $\boldsymbol{s} = (\boldsymbol{s}_l, \boldsymbol{s}_g)$ with $\boldsymbol{s}_l \in \mathbb{Z}^d$ and $\boldsymbol{s}_g \in \mathbb{Z}^r$ where $d := m + n + 1 - r$. This represents the short vector $\boldsymbol{v}$ by $(\boldsymbol{v}_l, \boldsymbol{v}_g)$ where $\boldsymbol{v}_l = (\boldsymbol{e}, 1, \boldsymbol{s}_l) \in \mathbb{Z}^d$ and $\boldsymbol{v}_g = \boldsymbol{s}_g$ with $\mathsf{HW}(\boldsymbol{v}_g) \leqslant h$.

Now one can simply apply Algorithm 9 with $h_M = \lfloor h/2 \rfloor$, but it takes enormous time for the most of our interest parameters. Instead, we pick smaller h_M to have feasible MitM cost, while expecting $\boldsymbol{s}_g$ has smaller weight. Since this naturally introduces some chance that algorithm fails, this parameter h_M would be appropriately chosen to minimize the overall complexity by considering the failure probability. We deal with this probability below by p_{h_M} in Lemma 5.4.2. The detailed algorithm can be found in Algorithm 10.

Adapting Scaling Factor

We also adapt the scaling factor technique [BG14] to our case. Precisely, we use the following basis

$$B_\nu = \left(\begin{array}{ccc|c} qI_m & & * & * \\ & \nu I_{n+1-r} & & 0 \\ \hline 0 & & 0 & I_r \end{array}\right)$$

that contains a vector $(\boldsymbol{v}'_l, \boldsymbol{v}_g)$ with $\boldsymbol{v}'_l = (\boldsymbol{e}, \nu, \nu\boldsymbol{s}_l)$ and $\boldsymbol{v}_g = \boldsymbol{s}_g$. The scaling factor ν is chosen to satisfy $\|\boldsymbol{v}'_l\| \approx \frac{\alpha q}{\sqrt{2\pi}}\sqrt{d}$ in order to assume $\boldsymbol{v}'_l$ as a vector sampled from discrete Gaussian $\mathcal{D}^d_{\alpha q}$. The explicit formula is given by

$$\nu = \frac{\alpha q}{\sqrt{2\pi}} \cdot \sqrt{\frac{n+1-r}{h+1-\mathsf{HW}(\boldsymbol{s}_l)}}.$$

Algorithm 10: A Primal Hybrid Attack

Input : $\mathsf{LWE}_{n,q,\alpha}(\mathcal{B}_h)$ sample $(A, \boldsymbol{b}) \in \mathbb{Z}_q^{m\times(n+1)}$
A blocksize β
MitM dimension parameter r
MitM weight parameter h_M

Output : LWE secret vector $\boldsymbol{s} \in \{\pm 1, 0\}^n$

1 $\nu \leftarrow \frac{\alpha q}{\sqrt{2\pi}} \cdot \sqrt{\frac{n+1}{h-2h_M+1}}$;
2 $y \leftarrow 6\alpha q/\sqrt{2\pi}$ `//  According to Assumption 5.4.1`;
3 Parse $A' = [-\boldsymbol{b} \mid A]$ into $[A'_1|A'_2]$ where A'_2 has r columns;
4 $B_\nu \leftarrow \begin{pmatrix} T & C \\ 0 & I_r \end{pmatrix}$ where $T = \begin{pmatrix} qI_m & A'_1 \\ 0 & I_{n+1-r} \end{pmatrix}$ and $C = \begin{pmatrix} A'_2 \\ 0 \end{pmatrix}$;
5 Run Algorithm 9 on input B_ν, β, h_M, y.

Versus the previous primal attack model

We give a brief intuition that explains how the primal hybrid attack performs better than the previous primal attack model. Recall from Section 5.4.1, the previous model takes advantage of sparsity by reducing the dimension of LWE by removing some columns of A, while expecting all the removed columns correspond to zero components of the secret. In our view of dividing

$$B = \left(\begin{array}{cc|c} qI_m & * & * \\ 0 & I_{n+1-r} & 0 \\ \hline 0 & 0 & I_r \end{array}\right) \text{ and } \boldsymbol{v} = \begin{pmatrix} \boldsymbol{v}_l \\ \boldsymbol{v}_g \end{pmatrix},$$

this translates into expecting the vector $\boldsymbol{v}_g = \boldsymbol{s}_g$ is zero, and apply the lattice reduction only for the upper-left matrix. Then the success probability is calculated by the probability that $\boldsymbol{s}_g = \mathbf{0}$. In this regard, our hybrid attack can be viewed to admit some nonzero components on $\boldsymbol{v}_g$ as long as the cost for investigating them remains not so large, which results in larger success probability.

5.4.4 Complexity Analysis

In this section we complete the analysis of hybrid attacks in Section 12 by calculating the probabilities with respect to parameters d, r and so on. We remark that although overall flow of analysis is similar to previous works for hybrid attacks [HG07, BGPW16, Wun16], but to the best of our knowledge, our analysis based on the MitM weight parameter h_M and Gaussian shape of $\boldsymbol{v}_l$ has never been considered before.

Recall that we defined

$$W = \{\boldsymbol{w} \in \{\pm 1, 0\}^r : \mathsf{HW}(\boldsymbol{w}) = h_M\}$$

and

$$V = \{\boldsymbol{w} \in W : (\boldsymbol{v}_g - \boldsymbol{w} \in W) \wedge (\mathsf{NP}_T(C\boldsymbol{w}) + \mathsf{NP}(C\boldsymbol{v}_g - C\boldsymbol{w}) = \boldsymbol{v}_l)\},$$

and two probabilities

$$p_s := \Pr_{\substack{\boldsymbol{w} \leftarrow W \\ \boldsymbol{v}_l \leftarrow \mathcal{D}^d_{\alpha q}}} [\mathsf{NP}_T(C\boldsymbol{w}) + \mathsf{NP}_T(C\boldsymbol{v}_g - C\boldsymbol{w}) = \boldsymbol{v}_l] \tag{5.8}$$

and

$$p_c := \Pr_{\boldsymbol{w} \leftarrow W}[\boldsymbol{v}_g - \boldsymbol{w} \in W]. \tag{5.9}$$

Now we will calculate the probabilities as following:

- Lemma 5.4.2 calculates the probability p_c under the assumption $\mathsf{HW}(\boldsymbol{v}_g) = 2k$ for some $k \leqslant h_M$ of probability p_{h_M}
- Lemma 5.4.3 calculates the probability p_s under the assumption $\mathsf{NP}_T(C\boldsymbol{v}_g) = \boldsymbol{v}_l$ of probability p_{NP}.

Then finally we fully represent T_{BKZ} and T_{guess} with regard to n, q, α, h and β, r, h_M, m and we finally conclude the total complexity estimation

$$T_{tot} = \frac{1}{p_{\mathsf{NP}} p_{h_M}} \left(T_{BKZ} + T_{guess}\right). \tag{5.10}$$

Lemma 5.4.2. *Let $\boldsymbol{v}_g \in \mathbb{Z}^r$ be a vector obtained by picking r components of vector $\boldsymbol{v}$ sampled uniformly from $\mathcal{B}_{n,h}$. Then the probability p_{h_M}*

of $\mathsf{HW}(\boldsymbol{v}_g) = 2k$ *for some* $k \leqslant h_M$ *is*

$$p_{h_M} = \sum_{k=0}^{h_M} \frac{\binom{h}{2k} \cdot \binom{n-h}{r-2k}}{\binom{n}{r}}.$$

Moreover, conditioned on $\mathsf{HW}(\boldsymbol{v}_g) = 2k$ *for some* $k \leqslant h_M$*, the probability* p_c *defined as (5.9) is represented by*

$$p_c = \sum_{k=0}^{h_M} \frac{1}{2^k} \frac{\binom{2k}{k}\binom{r-2k}{h_M-k}}{\binom{r}{h_M}} \cdot \frac{\binom{h}{2k}\binom{n-h}{r-2k}}{\sum_{i=0}^{h_M}\binom{h}{2i}\binom{n-h}{r-2i}}.$$

Proof. The probability p_{h_M} can be directly obtained from

$$\Pr[\mathsf{HW}(\boldsymbol{v}_g) = 2k] = \frac{\binom{h}{2k}\binom{n-h}{r-2k}}{\binom{n}{r}}.$$

For p_c, we write E be the event $\mathsf{HW}(\boldsymbol{v}_g) = 2k$ for some $k \leqslant h_M$, and split p_c by the conditional probabilities

$$p_c = \sum_{k=0}^{h_M} \Pr_{\boldsymbol{w} \leftarrow W}\left[\boldsymbol{v}_g - \boldsymbol{w} \in W \mid \mathsf{HW}(\boldsymbol{v}_g) = 2k\right] \cdot \Pr\left[\mathsf{HW}(\boldsymbol{v}_g) = 2k \mid E\right].$$

The latter probability is easily obtained by

$$\Pr\left[\mathsf{HW}(\boldsymbol{v}_g) = 2k \mid E\right] = \frac{\binom{h}{2k}\binom{n-h}{r-2k}}{\sum_{i=0}^{h_M}\binom{h}{2i}\binom{n-h}{r-2i}},$$

and we proceed to compute

$$\Pr_{\boldsymbol{w} \leftarrow W}\left[\boldsymbol{v}_g - \boldsymbol{w} \in W \mid \mathsf{HW}(\boldsymbol{v}_g) = 2k\right].$$

For that we observe, in order that $\boldsymbol{v}_g - \boldsymbol{w} \in \{\pm 1, 0\}^r$, $\boldsymbol{w}$ and $\boldsymbol{v}_g$ should agree on every position where $\boldsymbol{w}$ and $\boldsymbol{v}_g$ are both nonzero; if not, $\boldsymbol{v}_g - \boldsymbol{w}$ contains entry 2 or -2. By writing the number of such coincident components by ℓ, we have

$$\mathsf{HW}(\boldsymbol{v}_g - \boldsymbol{w}) = 2k - \ell + (h_M - \ell),$$

and ℓ should be k in order to have $\mathsf{HW}(\boldsymbol{v}_g - \boldsymbol{w}) = h_M$. Therefore, $\boldsymbol{w}$ should coincide with $\boldsymbol{v}_g$ exactly on k nonzero components for $\mathsf{HW}(\boldsymbol{v}_g - \boldsymbol{w}) = h_M$, from which we have

$$\Pr_{\boldsymbol{w} \leftarrow W}\left[\boldsymbol{v}_g - \boldsymbol{w} \in W \mid \mathsf{HW}(\boldsymbol{v}_g) = 2k\right] = \frac{1}{2^k} \frac{\binom{2k}{k}\binom{r-2k}{h_M - k}}{\binom{r}{h_M}}.$$

□

To proceed to the probability p_s and p_{NP} related to nearest plane algorithm, we require the following assumption.

Assumption 5.4.3. *We assume that the distribution of*

$$C\boldsymbol{w} \mod \mathcal{P}(T^*)$$

for $\boldsymbol{w} \leftarrow W$ is sufficiently close to the uniform distribution on $\mathcal{P}(T^*)$. Moreover, we assume that the discrete Gaussian $\mathcal{D}_{\alpha q}$ behaves like a continuous Gaussian distribution of standard deviation $\alpha q/\sqrt{2\pi}$.

Explanation. The first claim of this assumption has not been exactly stated in any previous analysis, but all of them also explicitly assumed this. For this to be plausible, it would be better to run Algorithm 9 with

$$T' = \begin{pmatrix} * & qI_m \\ \nu I_{n+1-r} & 0 \end{pmatrix},$$

which perturbs the coordinate axes determined by T'^* away from the standard coordinate axes of $C\boldsymbol{w}$. However, for brevity, we just put this by assumption instead of giving too much detail on this. □

Lemma 5.4.3. *Let R_i be the i-th Gram-Schmidt norm of T, and let $\boldsymbol{v}_l$ be a vector sampled from $\mathcal{D}^d_{\alpha q}$. Provided with Assumption 5.4.3, the probability p_{NP} of $\mathsf{NP}_T(\boldsymbol{v}_l) = \boldsymbol{v}_l$ is*

$$p_{NP} = \prod_{i=1}^{d} \operatorname{erf}\left(\frac{R_i\sqrt{\pi}}{2\alpha q}\right).$$

Moreover, conditioned on $\mathsf{NP}_T(\boldsymbol{v}_l) = \boldsymbol{v}_l$, we can represent the probability p_s defined as (5.8) by

$$p_s = \prod_{i=1}^{d}\left(\operatorname{erf}\left(\frac{R_i\sqrt{\pi}}{\alpha q}\right) + \frac{\alpha q}{R_i}\cdot\frac{e^{-\left(\frac{R_i\sqrt{\pi}}{\alpha q}\right)^2}-1}{\pi}\right).$$

Proof. For readability, we denote $\sigma := \alpha q/\sqrt{2\pi}$. We first compute the probability for $\mathsf{NP}_T(C\boldsymbol{v}_g) = \boldsymbol{v}_l$, or $\mathsf{NP}_T(\boldsymbol{v}_l) = \boldsymbol{v}_l$. By Lemma 2.2.1, this is equivalent to $\boldsymbol{v}_l \in \mathcal{P}(T^*)$. We assume that $\mathcal{D}^d_{\alpha q}$ is invariant to coordinate axes, we may assume that $\boldsymbol{v}_l$ is sampled with respect to the coordinate axes determined by T^*. Then we have

$$\begin{aligned}\Pr_{\boldsymbol{v}_l\leftarrow\mathcal{D}^d_\sigma}[\boldsymbol{v}_l \in \mathcal{P}(T^*)] &= \prod_{i=1}^{d}\Pr_{e\leftarrow\mathcal{D}_\sigma}[-R_i/2 \leqslant e \leqslant R_i/2] \\ &= \prod_{i=1}^{d}\operatorname{erf}\left(\frac{R_i}{2\sqrt{2}\sigma}\right).\end{aligned}$$

Toward p_s, we first show that

$$\mathsf{NP}_T(C\boldsymbol{w}) + \mathsf{NP}_T(C\boldsymbol{v}_g - C\boldsymbol{w}) = \boldsymbol{v}_l$$

is equivalent to

$$\mathsf{NP}_T(C\boldsymbol{w}) - \boldsymbol{v}_l \in \mathcal{P}(T^*).$$

Since our assumption says $\boldsymbol{v}_l = \mathsf{NP}_T(C\boldsymbol{v}_l) = \mathsf{NP}_T(C\boldsymbol{v}_g)$, and hence we only need to show that

$$\mathsf{NP}_T(C\boldsymbol{w}) + \mathsf{NP}_T(C\boldsymbol{v}_g - C\boldsymbol{w}) = \mathsf{NP}_T(C\boldsymbol{v}_g)$$

is equivalent to

$$\mathsf{NP}_T(C\boldsymbol{w}) - \mathsf{NP}_T(C\boldsymbol{v}_g) \in \mathcal{P}(T^*) :$$

Since $\mathsf{NP}_T(C\boldsymbol{v}_g - C\boldsymbol{w})$ belongs to $\mathcal{P}(T^*)$ by definition, the forward case directly holds. The reverse case also immediately holds because

$$\mathsf{NP}_T(C\boldsymbol{w}) - \mathsf{NP}_T(C\boldsymbol{v}_g) = -\mathsf{NP}_T(C\boldsymbol{v}_g - C\boldsymbol{w}) + T\boldsymbol{x}$$

for some $\boldsymbol{x}$.

Then we can represent

$$\begin{aligned} p_s &= \Pr_{\substack{\boldsymbol{t}\leftarrow\mathcal{P}(T^*)\\ \boldsymbol{e}\leftarrow\mathcal{D}_\sigma^d}}[\boldsymbol{t}+\boldsymbol{e} \in \mathcal{P}(T^*)] \\ &= \prod_{i=1}^{d} \Pr_{\substack{t\leftarrow[-R_i/2,R_i/2]\\ e\leftarrow\mathcal{D}_\sigma}}[t+e \in [-R_i/2, R_i/2]]\,. \end{aligned}$$

We now calculate $p_i := \Pr[-R_i/2 \leqslant t+e \leqslant R_i/2]$. Let $g(z)$ be the probability density function of $t+e$, which can be represented by probability

convolution

$$\begin{aligned} g(z) &= \frac{1}{R_i} \cdot \Pr_{e \leftarrow D_\sigma}[z - R_i/2 \leqslant e \leqslant z + R_i/2] \\ &= \frac{1}{2R_i} \cdot \left(\mathrm{erf}\left(\frac{z + R_i/2}{\sqrt{2}\sigma} \right) - \mathrm{erf}\left(\frac{z - R_i/2}{\sqrt{2}\sigma} \right) \right). \end{aligned}$$

Using the fact $\int \mathrm{erf}(x)dx = x \cdot \mathrm{erf}(x) + \frac{e^{-x^2}}{\sqrt{\pi}} + C$, we reach

$$\begin{aligned} p_i &= \int_{-R_i/2}^{R_i/2} g(z)dz \\ &= \frac{1}{2R_i} \cdot \int_{-R_i/2}^{R_i/2} \mathrm{erf}\left(\frac{z + R_i/2}{\sqrt{2}\sigma} \right) - \mathrm{erf}\left(\frac{z - R_i/2}{\sqrt{2}\sigma} \right) dz \\ &= \mathrm{erf}\left(\frac{R_i}{\sqrt{2}\sigma} \right) + \frac{\sqrt{2}\sigma}{R_i} \cdot \frac{e^{-\frac{R_i^2}{2\sigma^2}} - 1}{\sqrt{\pi}}. \end{aligned}$$

□

5.5 Bit-security estimation

In this section, we estimate the bit-security of LWE with small and sparse secret. Given LWE parameters n, q, α, h we choose optimal algorithm parameters β, r, h_M, m so that the total cost (5.10)

$$T_{tot} = \frac{1}{p_{\mathsf{NP}} p_{h_M}} \left(T_{BKZ} + T_{guess} \right).$$

is minimized, which determines the bit-security of given LWE parameters. The optimal parameters can be found by investigating possible choices for β, r, h_M, m, and we implement a `Sage` module that finds the (semi-)optimal

parameters**.

We stress again that, our analysis for the MitM hybrid attack is valid only when it holds that $|W| \geqslant \frac{1}{p_c p_s}$ regarding Assumption 5.4.2. For the parameters where the opposite case occurs, we estimate the cost with exhaustive search method by Section 5.5

Exhaustive-search hybrid-primal

Since the reduction cost is exactly same to Algorithm 9, it only suffices to clarify the guessing cost T_{guess}, which was estimated by $L \cdot T_{\mathsf{NP}}$ with Assumption 5.4.1 where L is the expected number of loops. For Algorithm 8 with weight parameter h_M, we simply upper bound L by $|W| = 2^{h_M}\binom{r}{h_M}$. Moreover, one can easily check that a sufficient condition for Algorithm 8 success is $\mathsf{NP}_T(\boldsymbol{v}_l) = \boldsymbol{v}_l$ and $\mathsf{HW}(\boldsymbol{v}_g) = h_g$, whose probabilities are denoted by p_{NP} and p_{h_g}. Note that p_{NP} is already computed by Lemma 5.4.3, and p_{h_g} can be easily computed by

$$p_{h_g} = \frac{\binom{h}{h_g}\binom{n-h}{r-h_g}}{\binom{n}{r}}.$$

Putting together everything, we conclude the total complexity of Algorithm 8 by

$$\frac{1}{p_{\mathsf{NP}} p_{h_g}} \left(T_{BKZ} + T_{guess}\right). \tag{5.11}$$

where $T_{guess} = 2^{h_M}\binom{r}{h_M} \cdot d^2/2^{1.06}$.

**The optimal parameters can be found by brutally searching all possible choices for β, r, h_M, m but there are too many candidates and hence estimation itself takes too much time. In this regard, we only investigate a plausible range of parameter sets to quickly see the cost estimation, while assuming the optimal point is indeed in our searching scope.

5.5.1 Estimations

Current implementations of HElib (commit `5bcae5f`) and HEAAN (commit `b45d5f0`) are commonly set sparse ternary secret of Hamming weight $h = 64$, and the noise parameter $\alpha = 8/q$ (yielding standard deviation $\sigma \approx 3.2$). HE-based applications built upon the libraries also use the setting and adjust dimension n and modulus q to reach the desired security level; for example [TLW+19, CHK+18, CCS19]. Thus we estimate attack complexity with the prevalent values for $h = 64$ and $\alpha = 8/q$, for several choices of n and q. We present Table 5.4 obtained by assuming sieving method for core SVP oracle.

n	1024	2048	4096	8192	16384	32768	65536
$\log q$	22	45	82	158	350	628	1240
Dual [Alb17]	129.3	127.7	129.5	128.6	128.3	127.2	130.3
Primal [AGVW17]	139.0	135.6	144.4	148.6	140.3	151.3	153.4
Hyb-Dual	**130.7**	**118.8**	**113.7**	**113.9**	**104.6**	**112.5**	**115.4**
Hyb-Primal	**100.9**	**96.7**	**102.1**	**104.9**	**101.8**	**109.3**	**112.9**

Table 5.4: Costs with $h = 64$ and $\alpha = 8/q$ (Sieving SVP oracle)

The both hybrid attacks show better performance than the current best attack (Albrecht's dual attack) for modulus $q \geqslant 2^{40}$, and hence our attacks claim that fully homomorphic encryption implementations that uses the sparse ternary LWE problem with large modulus q should change the parameter selection. In particular, HElib [HS19] and HEAAN [CHK+19] use the sparse ternary secret basically. SEAL [SEA19] uses the (non-sparse) ternary secret key but the paper [CH18] that supports bootstrapping for SEAL also uses the sparse ternary secret vector.

However, the hybrid-dual attack shows worse performance for small modulus. In this regard, we note that Albrecht's dual attack that can be

regarded as a special case of our hybrid-dual attack with $h_1 = 0$, and hence, if we investigate all possible parameter range in our code, our hybrid-dual algorithm must outperform Albrecht's dual attack. However it takes too much time to check all possible parameter ranges, and we instead investigate plausible range of parameters; our code only explores the parameter regime that $h_1, h_2 \gtrsim h/2$, and this may not capture the real optimal point. Meanwhile, the estimations for small modulus q size implies the exhaustive search is better than the MitM approach for that parameter, which seems weird at first glance. However this enough make sense because our MitM algorithm runtime exponentially grows with B/q, where B is the error size. Then, to have small B/q after the dimension-error trade-off, we may have to find shorter vectors in the lattice reduction stage than Albrecht's dual attack. Particularly for small modulus q, the additional cost for finding such shorter vector offsets the benefit of MitM approach.

5.5.2 Application to PKE

The round 2 candidates of NIST Post-Quantum Cryptography Standardization includes several lattice-based schemes, and we find one scheme named `Round5` [BBF+19] that uses sparse and ternary secret. The base problem of `Round5` is the *learning with rounding* (LWR) problem, defined in similar way to LWE problem with additional modulus $p < q$ and

$$\left(A, \left\lfloor \frac{p}{q} \cdot A\boldsymbol{s} \right\rceil\right) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_p^m$$

It can be viewed that the noise from the rounding plays the Gaussian error role of LWE. Indeed for the security estimation, LWR with modulus p and

q is understood by LWE with error having standard deviation

$$\sigma = \frac{q}{p} \cdot \frac{1}{\sqrt{12}},$$

and the typical LWE attacks are applied to estimate its bit-security.

We find that the authors already considered the hybrid attack to choose parameters while conservatively assuming BKZ_β cost, regardless of the dimension d of lattice, by

$$T_{BKZ}(\beta, d) = 2^{0.292\beta}$$

according to [ADPS16].

According to their analysis, the hybrid attack indeed shows the best performance for its parameter sets. In this regard, we briefly point out here some flaws and insufficiency of their analysis, However, they merely estimate the guessing cost T_{guess} by $\sqrt{N}$ where N is the expected number of candidates of secret vectors, which is quite improper to derive accurate time cost. Moreover, whereas our algorithm introduces a MitM weight parameter h_M to have a trade-off between the success probability and the guessing cost, they only consider the full cost for guessing every possible candidates. Taking this into account, we re-evaluate the bit-security of the proposed parameters according to our refined analysis, and hence conclude that the security of their parameter choice is overestimated

We first remark that, this inferiority of the hybrid attack for `Round5` is in line with the argument that the hybrid attack shows worse performance than previous thought for NTRU, which was stated by [Wun16]. Moreover, the ratio of Hamming weight to the dimension should also be noticed to understand this inferiority compared to HE; `Round5` has weight 162 out of

(Claimed to be) 128 bit-security				
n	$\log q$	h	σ	**Hybrid**
490	10	162	2.29	147.7
508	10	136	2.29	141.8
586	13	182	4.61	146.0
618	11	104	2.29	131.7

Table 5.5: Solving costs for LWR instances, which were claimed to have $\lambda = 128$ security level in [BBF+19], with BKZ cost model $2^{0.292\beta}$ [ADPS16].

490 (33%) while HEAAN has weight 64 out of from 2048 to 65536 (from 3% to 0.1%), and this may let combinatorial strategy of the hybrid attack bring larger performance gain for the extremely sparse secret of HE.

Chapter 6

Conclusion

In this paper we examine several requirements on the actual use of homomorphic encryption. First, we consider an ID-based scenario where data accessibility can be authorized by user's unique ID. In this regard, we design a new paradigm of ID-based homomorphic encryption where the data is first encrypted in plain ID-based ciphertext and then recrypted into homomorphic encryption. For this purpose, we also propose more efficient trapdoor-based ID-based encryption, where the hardness of trapdoor is based on Module-NTRU problem.

We also propose a fundamental solution for secret key management by proposing a new biometric key decryption method. Our proposal has polynomial performance in key error rate t, compared to the best previous result having exponential performance.

Finally, for a concrete implementation and evaluation, we rigorously examine the security of homomorphic encryption schemes. As a result, we propose new attack algorithms that show current parameter settings of homomorphic encryption cannot satisfy the claimed security level.

Bibliography

[ACC+18] Martin Albrecht, Melissa Chase, Hao Chen, Jintai Ding, Shafi Goldwasser, Sergey Gorbunov, Shai Halevi, Jeffrey Hoffstein, Kim Laine, Kristin Lauter, Satya Lokam, Daniele Micciancio, Dustin Moody, Travis Morrison, Amit Sahai, and Vinod Vaikuntanathan. Homomorphic encryption security standard. Technical report, HomomorphicEncryption.org, Toronto, Canada, November 2018.

[ACEK17] Daniel Apon, Chongwon Cho, Karim Eldefrawy, and Jonathan Katz. Efficient, reusable fuzzy extractors from lwe. In *International Conference on Cyber Security Cryptography and Machine Learning*, pages 1–18. Springer, 2017.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange-a new hope. In *USENIX Security Symposium*, volume 2016, 2016.

[AGVW17] Martin R Albrecht, Florian Göpfert, Fernando Virdia, and Thomas Wunderer. Revisiting the expected cost of solving usvp and applications to lwe. In *Proc. of ASIACRYPT '17*, pages 297–322. Springer, 2017.

[AL18] Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. *SIAM Journal on Computing*, 47(1):52–79, 2018.

[Alb17] Martin R Albrecht. On dual lattice attacks against small-secret LWE and parameter choices in helib and SEAL. In *Proc. of EUROCRYPT '17*, pages 103–129. Springer, 2017.

[App16] Benny Applebaum. Cryptographic hardness of random local functions. *Computational complexity*, 25(3):667–722, 2016.

[APS15] Martin R Albrecht, Rachel Player, and Sam Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[Bab86] László Babai. On lovász'lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[Ban95] Wojciech Banaszczyk. Inequalities for convex bodies and polar reciprocal lattices inr n. *Discrete & Computational Geometry*, 13(2):217–231, 1995.

[BBF+19] Hayo Baan, Sauvik Bhattacharya, Scott Fluhrer, Oscar Garcia-Morchon, Thijs Laarhoven, Ronald Rietman, Markku-Juhani O. Saarinen, Ludo Tolhuizen, and Zhenfei Zhang. Round5: Compact and fast post-quantum public-key encryption. In Jintai Ding and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 83–102, Cham, 2019. Springer International Publishing.

[BDGL16] Anja Becker, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *Proc. of SODA '16*, pages 10–24, 2016.

[BDK+18] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-kyber: a cca-secure module-lattice-based kem. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing. In *Annual international cryptology conference*, pages 213–229. Springer, 2001.

[BG14] Shi Bai and Steven D Galbraith. Lattice decoding attacks on binary lwe. In *Australasian Conference on Information Security and Privacy*, pages 322–337. Springer, 2014.

[BGPW16] Johannes Buchmann, Florian Göpfert, Rachel Player, and Thomas Wunderer. On the hardness of lwe with binary error: revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In *International Conference on Cryptology in Africa*, pages 24–43. Springer, 2016.

[Boy04] Xavier Boyen. Reusable cryptographic fuzzy extractors. In *Proceedings of the 11th ACM conference on Computer and communications security*, pages 82–91. ACM, 2004.

[BQ12] Andrej Bogdanov and Youming Qiao. On the security of goldreich's one-way function. *Computational complexity*, 21(1):83–127, 2012.

[BR13] Andrej Bogdanov and Alon Rosen. Input locality and hardness amplification. *Journal of cryptology*, 26(1):144–171, 2013.

[CCS19] Hao Chen, Ilaria Chillotti, and Yongsoo Song. Improved bootstrapping for approximate homomorphic encryption. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part II*, pages 34–54, 2019.

[CFP+16] Ran Canetti, Benjamin Fuller, Omer Paneth, Leonid Reyzin, and Adam Smith. Reusable fuzzy extractors for low-entropy distributions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 117–146. Springer, 2016.

[CH18] Hao Chen and Kyoohyung Han. Homomorphic lower digits removal and improved fhe bootstrapping. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 315–337. Springer, 2018.

[Che13] Yuanmi Chen. *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*. PhD thesis, Paris 7, 2013.

[CHHS19] Jung Hee Cheon, Minki Hhan, Seungwan Hong, and Yongha Son. A hybrid of dual and meet-in-the-middle attack on sparse and ternary secret lwe. *IEEE Access*, 7:89497–89506, 2019.

[CHK+18] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 360–384. Springer, 2018.

[CHK+19] Jung Hee Cheon, Kyoohyung Han, Andrey Kim, Miran Kim, and Yongsoo Song. Snucrypto HEAAN. `https://github.com/homenc/HElib`, 2019.

[CHS19] Jung Hee Cheon, Minki Hhan, and Yongha Son. Reusable fuzzy extractors from local functions. In submission, 2019.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *Proc. of ASIACRPYT'17*, pages 409–437. Springer, 2017.

[CKKS19] Jung Hee Cheon, Duhyeong Kim, Taechan Kim, and Yongha Son. A new trapdoor over module-ntru lattice and its application to id-based encryption. In submission, 2019.

[CM01] Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in nc 0. In *International Symposium on Mathematical Foundations of Computer Science*, pages 272–284. Springer, 2001.

[CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In *Annual Cryptology Conference*, pages 487–504. Springer, 2011.

[CN11] Yuanmi Chen and Phong Q Nguyen. Bkz 2.0: Better lattice security estimates. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 1–20. Springer, 2011.

[Dau09] John Daugman. How iris recognition works. In *The essential guide to image processing*, pages 715–739. Elsevier, 2009.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[DLP14] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over ntru lattices. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 22–41. Springer, 2014.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM journal on computing*, 38(1):97–139, 2008.

[FPV18] Vitaly Feldman, Will Perkins, and Santosh Vempala. On the complexity of random satisfiability problems with planted

solutions. *SIAM Journal on Computing*, 47(4):1294–1338, 2018.

[Gal13] Steven D Galbraith. Space-efficient variants of cryptosystems based on learning with errors, 2013. `https://www.math.auckland.ac.nz/~sgal018/compact-LWE.pdf`.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. of STOC '09*, pages 169–178. ACM, 2009.

[GHS12] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Annual Cryptology Conference*, pages 850–867. Springer, 2012.

[Gol00] Oded Goldreich. Candidate one-way functions based on expander graphs. *IACR Cryptology ePrint Archive*, 2000:63, 2000.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proc. of STOC '08*, pages 197–206. ACM, 2008.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Proc. of CRYPTO'13*, pages 75–92. Springer, 2013.

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against ntru. *Proc. of CRYPTO '07*, pages 150–169, 2007.

[HGSW03] Nick Howgrave-Graham, Joseph Silverman, and William Whyte. A meet-in-the-middle attack on an ntru private key, 07 2003.

[HHGP+03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H Silverman, and William Whyte. Ntrusign: Digital signatures using the ntru lattice. In *Cryptographers' Track at the RSA Conference*, pages 122–140. Springer, 2003.

[HHHGW09] Philip S Hirschhorn, Jeffrey Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing ntruencrypt parameters in light of combined lattice reduction and mitm approaches. In *International Conference on Applied Cryptography and Network Security*, pages 437–455. Springer, 2009.

[HS14] Shai Halevi and Victor Shoup. Algorithms in helib. In *Proc. of CRYPTO '14*. Springer Verlag, 2014.

[HS19] Shai Halevi and Victor Shoup. Helib. `https://github.com/homenc/HElib`, 2019.

[Kan87] Ravi Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of operations research*, 12(3):415–440, 1987.

[KLRW14] Patrick Koeberl, Jiangtao Li, Anand Rajan, and Wei Wu. Entropy loss in puf-based key generation schemes: The repetition code pitfall. In *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 44–49. IEEE, 2014.

[LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for lwe-based encryption. In *Proc. of CT-RSA' 11*, volume 65–58, pages 319–339. Springer, 2011.

[Mat93] Mitsuru Matsui. Linear cryptanalysis method for des cipher. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 386–397. Springer, 1993.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspectiv*. Springer, 2002.

[MR07] Daniele Micciancio and Oded Regev. Worst-case to average-case reductions based on gaussian measures. *SIAM Journal on Computing*, 37(1):267–302, 2007.

[MST06] Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ε-biased generators in nc0. *Random Structures & Algorithms*, 29(1):56–81, 2006.

[O'D03] Ryan William O'Donnell. *Computational applications of noise sensitivity*. PhD thesis, Massachusetts Institute of Technology, 2003.

[OW14] Ryan O'Donnell and David Witmer. Goldreich's prg: Evidence for near-optimal polynomial stretch. In *Proceedings of the 2014 IEEE 29th Conference on Computational Complexity*, pages 1–12. IEEE Computer Society, 2014.

[PAFZ19] Paul Kirchner Vadim Lyubashevsky Thomas Pornin Thomas Prest Thomas Ricosset Gregor Seiler William Whyte Pierre-Alain Fouque, Jeffrey Hoffstein and Zhenfei Zhang. Fal-

con: Fast-fourier lattice-based compact signatures over ntru. *Post-Quantum Cryptography Standardization Round2 Submissions*, 2019. `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/round-2-submissions`.

[PDG14] Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 353–370. Springer, 2014.

[PP19] Thomas Pornin and Thomas Prest. More efficient algorithms for the ntru key generation using the field norm. In *IACR International Workshop on Public Key Cryptography*, pages 504–533. Springer, 2019.

[Pre17] Thomas Prest. Sharper bounds in lattice-based cryptography using the rényi divergence. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 347–374. Springer, 2017.

[Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proc. of STOC '05*, pages 84–93. ACM, 2005.

[SC19] Yongha Son and Jung Hee Cheon. Revisiting the hybrid attack on sparse and ternary secret lwe. In *7th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, pages 11–20. ACM, 2019.

[SEA19] Microsoft SEAL (release 3.3). `https://github.com/Microsoft/SEAL`, 2019. Microsoft Research, Redmond, WA.

[She99] WILLIAM FLEETWOOD Sheppard. On the application of the theory of error to cases of normal distribution and normal correlation. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 192:101–531, 1899.

[TLW+19] Benjamin Hong Meng Tan, Hyung Tae Lee, Huaxiong Wang, Shu Qin Ren, and Khin Mi Mi Aung. Efficient private comparison queries over encrypted databases using fully homomorphic encryption with finite fields. Cryptology ePrint Archive, Report 2019/332, 2019. `https://eprint.iacr.org/2019/332`.

[WL18] Yunhua Wen and Shengli Liu. Robustly reusable fuzzy extractor from standard assumptions. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 459–489. Springer, 2018.

[WLG19] Yunhua Wen, Shengli Liu, and Dawu Gu. Generic constructions of robustly reusable fuzzy extractor. In *IACR International Workshop on Public Key Cryptography*, pages 349–378. Springer, 2019.

[Wun16] Thomas Wunderer. Revisiting the hybrid attack: Improved analysis and refined security estimates. Cryptology ePrint Archive, Report 2016/733, 2016. `https://eprint.iacr.org/2016/733`.

국문초록

클라우드 상의 데이터 분석 위임 시나리오는 동형암호의 가장 효과적인 응용 시나리오 중 하나이다. 그러나, 다양한 데이터 제공자와 분석결과 요구자가 존재하는 실제 현실의 모델에서는 기본적인 암복호화와 동형 연산 외에도 여전히 해결해야 할 과제들이 남아있는 실정이다. 본 학위논문에서는 이러한 모델에서 필요한 여러 요구사항들을 포착하고, 이에 대한 해결방안을 논하였다.

먼저, 기존의 알려진 동형 데이터 분석 솔루션들은 데이터 간의 층위나 수준을 고려하지 못한다는 점에 착안하여, 신원기반 암호와 동형암호를 결합하여 데이터 사이에 접근 권한을 설정하여 해당 데이터 사이의 연산을 허용하는 모델을 생각하였다. 또한 이 모델의 효율적인 동작을 위해서 동형암호 친화적인 신원기반 암호에 대하여 연구하였고, 기존에 알려진 NTRU 기반의 암호를 확장하여 module-NTRU 문제를 정의하고 이를 기반으로 한 신원기반 암호를 제안하였다.

둘째로, 동형암호의 복호화 과정에는 여전히 비밀키가 관여하고 있고, 따라서 비밀키 관리 문제가 남아있다는 점을 포착하였다. 이러한 점에서 생체정보를 활용할 수 있는 복호화 과정을 개발하여 해당 과정을 동형암호 복호화에 적용하였고, 이를 통해 암복호화와 동형 연산의 전 과정을 어느 곳에도 키가 저장되지 않은 상태로 수행할 수 있는 암호시스템을 제안하였다.

마지막으로, 동형암호의 구체적인 안전성 평가 방법을 고려하였다. 이를 위해 동형암호가 기반하고 있는 이른바 Learning With Errors (LWE) 문제의 실제적인 난해성을 면밀히 분석하였고, 그 결과 기존의 공격 알고리즘보다 평균적으로 1000배 이상 빠른 공격 알고리즘들을 개발하였다. 이를 통해 현재 사용하고 있는 동형암호 파라미터가 안전하지 않음을 보였고, 새로운 공격 알고리즘을 통한 파라미터 설정 방법에 대해서 논하였다.

주요어휘: 신원기반암호, 양자내성암호, 동형암호, 잡음키암호
학번: 2014-21208