Cryptography with Lattices

07D37042 Keita Xagawa
Supervisor: Keisuke Tanaka

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology

February 15, 2010

Contents

1 Introduction

1.1 Backgrounds

After the seminal paper by Diffie and Hellman [DH76], several cryptosystems based on the number-theoretic problems were proposed, such as the RSA encryption scheme [RSA78], the ElGamal encryption scheme [ElG85], and etc. They have succeeded in the real life and academic world. We have used them in the real life and have taught cryptography with exemplifying them.

Although there were several cryptosystems based on combinatorial problems, less attentions were payed on them than to number-theoretical ones. In my opinion, one of reasons is their fragile lives; several cryptosystems were cryptanalysed with the proposed parameters after a few years from their proposal. For example, the Merkle–Hellman "knapsack" encryption scheme [MH78] and their variants were soon cryptanalysed in the realistic parameters. The other of reasons is their inefficiency; the above attacks are against the proposed parameters and, hence, they need the larger parameter sets to bear the attacks. In addition, their inherent structures yield huge public keys, say the quadratic or cubic order of the security parameter.

This situation was changed externally by a new threat on number-theoretical cryptosystems, i.e., Shor's quantum attacks [Sho97]. These schemes were shed by the light and provided much attentions. After this new threat, many researchers have made strenuous efforts to construct secure schemes and found several combinatorial problems suiting to do.

An attractive one of them is lattice-based cryptography; it appeared in 1996 to construct one-way functions with average-case/worst-case equivalence in Ajtai [Ajt96], which rarely appears in number-theoretic problems and other combinatorial problems. Lattices have already appeared as the cryptanalytic tools in cryptography. See the surveys [Cai98a, NS01].

Lattice-based cryptography have bloomed in this two decades; we have obtained hash functions, digital signatures, public-key encryptions, identity-based primitives, and etc. and they enjoyed average-case/worst-case equivalences, that is, their securities are based on the worst-case hardness of certain lattice problems.

In this thesis, we will review them and give some intuitions on the constructions of them. The organization of this thesis is in Section 1.3. We first prepare the notions and notation.

1.2 Preliminaries

In this section, we review basic notions and notation on probabilities, distributions, hash functions, and protocols which will appear in this thesis.

1.2.1 Basic notions and notation

We define a negligible amount in n as an amount that is asymptotically smaller than n^{-c} for any constant $c > 0$. More formally, We say a function $f(n)$ is a negligible function in n if $\lim_{n\to\infty} n^c f(n) = 0$ for any $c > 0$. Similarly, a non-negligible amount is one which is at least n^{-c} for some $c > 0$. We denote by n the security parameter of cryptographic schemes throughout this paper, which corresponds to the rank of the underlying lattice problems. We say that a problem is hard in the worst case if there exists no probabilistic polynomial-time algorithm solves the problem in the worst case with non-negligible probability. We sometimes use $\tilde{O}(g(n))$ for any function g in n as $O(g(n) \cdot \mathrm{polylog}(g(n)))$. We assume that all random variables are independent and uniform. For a positive integer n, let $[n]$ denote a set $\{1, 2, \ldots, n\}$.

Vectors will be denoted by bold italic, say $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$, etc. Polynomials are denoted by bold roman, say $\mathbf{a}, \mathbf{b}, \mathbf{c}$. In addition, we denote vectors of polynomials by bold italic with a check, $\check{\boldsymbol{a}}, \check{\boldsymbol{b}}, \check{\boldsymbol{c}}$, etc. We denote matrices by upper bold italic such as $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$.

To denote a column vector with elements, we write elements in parentheses; $\boldsymbol{a} = (a_1, \ldots, a_m)$. If a row vector, we denote by $[a_1, \ldots, a_m]$, elements in brackets. We often compose a matrix. If we write $\boldsymbol{A} = [\boldsymbol{A}_1 | \boldsymbol{A}_2]$ with $\boldsymbol{A}_1 \in S^{n \times m}$ and $\boldsymbol{A}_2 \in S^{n \times l}$, $\boldsymbol{A}$ is an n by $(m + l)$ matrix. If we write $\boldsymbol{A} = [\boldsymbol{A}_1; \boldsymbol{A}_2]$ where $\boldsymbol{A}_1$ is an n by m and $\boldsymbol{A}_2$ is an l by m then $\boldsymbol{A}$ is an $(n + l)$ by m matrix.

For any $p \geq 1$, the l_p norm of a vector $\boldsymbol{x} = {}^t(x_1, \ldots, x_n) \in \mathbb{R}^n$, denoted by $\|\boldsymbol{x}\|_p$, is $(\sum_{i \in [n]} x_i^p)^{1/p}$. For ease of notation, we define $\|\boldsymbol{x}\| := \|\boldsymbol{x}\|_2$. The l_∞ norm is defined as $\|\boldsymbol{x}\|_\infty = \lim_{p\to\infty} \|\boldsymbol{x}\|_p = \max_{i \in [n]} |x_i|$. Let $w_H(\boldsymbol{x})$ denote the Hamming weight of $\boldsymbol{x}$, i.e., the number of non-zero elements in $\boldsymbol{x}$. Let $\mathcal{S}(m, w)$ denote the set of binary vectors in $\{0, 1\}^m$ whose Hamming weights are exactly equal to w, i.e., $\mathcal{S}(m, w) := \{\boldsymbol{x} \in \{0, 1\}^m \mid w_H(\boldsymbol{x}) = w\}$. We denote the concatenation of two vectors or strings $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ by $\boldsymbol{v}_1 \circ \boldsymbol{v}_2$.

$B_n^p(\boldsymbol{c}, r)$ denotes an n-dimensional ball centered $\boldsymbol{c} \in \mathbb{R}^n$ and with radius $r \geq 0$

in the l_p norm. We drop n if the dimension n is not ambiguous in the context. We drop p if $p = 2$ and drop $\boldsymbol{c}$ if the center is the origin, that is, $\boldsymbol{c} = \boldsymbol{0}$.

1.2.2 Probabilities and Distributions

Let ϕ_1 and ϕ_2 be two probability density functions on a finite set S. We often let ϕ_1 indicate a distribution corresponding to probability density function ϕ_1, vice verse. We define the statistical distance between two distributions ϕ_1 and ϕ_2 as $\Delta(\phi_1, \phi_2) := \frac{1}{2}\sum_{x \in S} |\phi_1(x) - \phi_2(x)|$. Given two distributions ϕ_1 and ϕ_2 over $\mathbb{R}^m$, which are continuous, we define the statistical distance between them as $\Delta(\phi_1, \phi_2) := \frac{1}{2}\sum_{\boldsymbol{x} \in \mathbb{R}^m} |\phi_1(x) - \phi_2(x)| dx$. We also use the same notation for two arbitrary functions. Note that the acceptance probability of any algorithm on inputs from X differs from its acceptance probability on inputs from Y by at most $\Delta(X, Y)$.

If $\mathsf{A}(\cdot, \cdot, \dots)$ is a randomized algorithm, then $y \leftarrow \mathsf{A}(x_1, x_2, \dots; r)$ means that y is assigned the unique output of the algorithm on inputs $x_1, x_2, \dots$ and coins r. We often use the notation $y \leftarrow \mathsf{A}(x_1, x_2, \dots)$ as shorthand for first picking r at random and then setting $y \leftarrow \mathsf{A}(x_1, x_2, \dots, ; r)$. If S is a finite set then $s \leftarrow S$ indicates that s is chosen uniformly at random from S. If D is a distribution then $x \leftarrow D$ indicates that x is chosen according to the distribution D.

We say two distributions D_1 and D_2 are *perfectly indistinguishable* if $D_1 = D_2$, denoted by $D_1 \approx_P D_2$. They are *statistically indistinguishable* if $\Delta(D_1, D_2)$ is negligible in the security parameter n, that is, $\Delta(D_1, D_2) \le n^{-\omega(1)}$. We denote them by $D_1 \approx_S D_2$. They are *computationally indistinguishable* if for any polynomial-time algorithm $\mathcal{A}$,

$$\left|\Pr[\mathcal{A}(1^n, X_1) = 1] - \Pr[\mathcal{A}(1^n, X_2) = 1]\right| \le n^{-\omega(1)},$$

where X_i is a random variable distributed according to D_1 for $i = 1, 2$ and the probabilities are taken by X, Y, and coins of $\mathcal{A}$. We denote them by $D_1 \approx_C D_2$.

Let X be a random variable over a set S. The *min-entropy* of X is defined by

$$H_\infty(X) = -\log \max_{x \in S} \Pr[X = x].$$

If $H_\infty(X)$ is $\log |\mathcal{S}|$, X is distributed uniformly over $\mathcal{S}$.

1.3 Organization

Chapter 2 reviews lattices, lattice problems, and relations and reductions among the problems. Chapter 3 also reviews ideal lattices, and more. In Chapter 4, we review hash functions based lattice problems and ideal lattice problems. In Chapter 5, we introduce simple string commitment schemes based on lattice problems. Chapter 6 gives two identification schemes which are variants of Stern's identification schemes and based on lattice and ideal lattice problems. Based on these two schemes, we construct ad hoc identification schemes in Chapter 8. As an interlude, we point out that Stern's scheme yields zero-knowledge and proof-of-knowledge protocols for NTRU in Chapter 9. Chapter 10 summarizes trapdoor generation algorithms by Ajtai and by Alwen and Peikert and propose ideal versions of the trapdoor generation algorithms. Chapter 11 reviews the signature scheme by Gentry, Peikert, and Vaikuntanathan, which employs the above trapdoor generation algorithms, and constructs a compact signature scheme following them. Combining the signature schemes and the Micciancio-Vadhan identification scheme, we obtain two identity-based identification scheme in Chapter 7. Survey on public-key encryption, key-encapsulation mechanism, and identity-based encryption schemes based on lattice problems appear in Chapter 12, Chapter 13, and Chapter 14, respectively. Chapter 15 proposes new lattice-based proxy re-encryption schemes, which are based on several encryption schemes.

2 Lattices

Organization: Section 2.1 give the basic definitions and notions on lattices. Section 2.2 reviews the problems on lattice appeared in the literature. In Section 2.3, we briefly review the results on the hardness of lattice problems. In Section 2.4, we give the review of the average-case/worst-case reductions.

2.1 Lattices

We first review fundamental notions of lattices.

A *lattice* is a discrete additive subgroup of $\mathbb{R}^m$. Formally, an n-dimensional lattice Λ in $\mathbb{R}^m$ is the set $\mathcal{L}(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n) = \{\sum_{i \in [n]} \alpha_i \boldsymbol{b}_i \mid \alpha_i \in \mathbb{Z}\}$ of all integral combinations of n linearly independent vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \in \mathbb{R}^m$. The sequence of vectors $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$ is called a *basis* of the lattice $\Lambda = \mathcal{L}(\boldsymbol{B})$ and denoted by $\boldsymbol{B} = [\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n]$, where $\boldsymbol{B}$ is an m by n matrix. Using this notation, we can write $\Lambda = \{\boldsymbol{Bx} \mid \boldsymbol{x} \in \mathbb{Z}^n\}$. Notice that a lattice has infinitely many bases. This can be confirmed by checking that $\boldsymbol{BU}$ is also a basis of L for any unimodular matrix $\boldsymbol{U} \in \mathbb{Z}^{n \times n}$, which is a matrix with determinant -1 or 1. In this thesis, we only consider the full-rank lattices, an n-dimensional lattice in $\mathbb{R}^n$.

The *dual* lattice of Λ, denoted by Λ^*, is $\Lambda^* = \{\boldsymbol{x} \in \mathbb{R}^n : \forall \boldsymbol{v} \in \Lambda, \langle \boldsymbol{x}, \boldsymbol{v} \rangle \in \mathbb{Z}\}$. It can be verified that $(\Lambda^*)^* = \Lambda$. If $\boldsymbol{B}$ is a basis of Λ, then the basis of the dual lattice is $\boldsymbol{B}^* = (\boldsymbol{B}^{-1})^T$.

For any set $\boldsymbol{S} = \{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\} \subset \mathbb{R}^n$ of linearly independent vectors, let $\tilde{\boldsymbol{S}} = \{\tilde{\boldsymbol{s}}_1, \ldots, \tilde{\boldsymbol{s}}_n\}$ be its Gram-Schmidt orthogonalization: for each $i \in [n]$,

$$\tilde{\boldsymbol{s}}_i = \begin{cases} \boldsymbol{s}_1, & \text{if } i = 1, \\ \boldsymbol{s}_i - \sum_{j \in [i-1]} \frac{\langle \boldsymbol{s}_i, \tilde{\boldsymbol{s}}_j \rangle}{\langle \tilde{\boldsymbol{s}}_j, \tilde{\boldsymbol{s}}_j \rangle} \tilde{\boldsymbol{s}}_j, & \text{otherwise} \end{cases}.$$

Note that $\|\tilde{\boldsymbol{s}}_i\| \le \|\boldsymbol{s}_i\|$ for any $i \in [n]$.

In the l_2 norm, for any full-rank set $\boldsymbol{S} \subset \Lambda$, there is a basis $\boldsymbol{T}$ of Λ such that $\|\tilde{\boldsymbol{T}}\| \le \|\tilde{\boldsymbol{S}}\| \le \|\boldsymbol{S}\|$.

Lemma 2.1.1 (Lemma 7.1, page 129, [MG02]). *There is a deterministic polynomial-time algorithm* MGReduce *that, given an arbitrary basis $\boldsymbol{B}$ of an n-dimensional lattice Λ and a full-rank set of lattice vectors $\boldsymbol{S} \subset \Lambda$, outputs a basis $\boldsymbol{T}$ of Λ such that $\|\tilde{\boldsymbol{t}}_i\| \le \|\tilde{\boldsymbol{s}}_i\|$ for all $i \in [n]$.*

Additionally, the Gram-Schmidt orthogonalization of a basis and its dual are closely related.

Lemma 2.1.2 ([Reg04a, Lecture 8]). *Let $\{\boldsymbol{b}_1, \dots, \boldsymbol{b}_n\}$ be an basis of Λ and let $\{\boldsymbol{d}_1, \dots, \boldsymbol{d}_n\}$ be its dual basis in reversed order ($\boldsymbol{d}_i = \boldsymbol{b}^*_{n-i+1}$). Then $\tilde{\boldsymbol{d}}_i = \tilde{\boldsymbol{b}}_i / \left\|\tilde{\boldsymbol{b}}_i\right\|^2$ for all $i \in [n]$.*

For more details on lattices, see the textbook by Micciancio and Goldwasser [MG02].

2.1.1 Lattice Constants

There are several constants for lattices which are independent of representations. The most fundamental one is the length of the shortest vector. This is generalized as successive minima $\lambda_i^p(\Lambda)$ for $i \in [n]$: For every i, the i-th minimum $\lambda_i^p(\Lambda)$ is the radius of the smallest sphere centered in the origin containing i linearly independent lattice vectors, that is,

$$\lambda_i^p(\Lambda) = \min\{r : \dim(\mathrm{span}(\Lambda \cap B^p(r))) \ge i\}.$$

Setting $i = 1$, $\lambda_1^p(\Lambda)$ stands for the length of the shortest vector in the lattice Λ in the l_p norm.

The definition of the covering radius $\mu(\Lambda)$ is given by

$$\mu(\Lambda) = \max_{\boldsymbol{x}_p \in \mathbb{R}^n} \{\mathrm{dist}(\boldsymbol{x}, \Lambda)\}.$$

The name is from the fact that $\bigcup_{\boldsymbol{v} \in \Lambda} B^p(\boldsymbol{v}, \mu(\Lambda)) = \mathbb{R}^n$.

Another lattice constant is the length of the shortest basis of Λ, $bl^p(\Lambda)$. This is defined as

$$bl^p(\Lambda) = \min_{\boldsymbol{B}:\text{a basis of }\Lambda} \|\boldsymbol{B}\|_p = \min_{\boldsymbol{B}:\text{a basis of }\Lambda} \max_{i \in [n]} \|\boldsymbol{b}_i\|.$$

In addition, we can define the Gram-Schmidt minimum as

$$\widetilde{bl}^p(\Lambda) = \min_{\boldsymbol{B}:\text{a basis of }\Lambda} \|\tilde{\boldsymbol{B}}\|_p = \min_{\boldsymbol{B}:\text{a basis of }\Lambda} \max_{i \in [n]} \|\tilde{\boldsymbol{b}}_i\|.$$

This constant is introduced explicitly in [GPV08] and implicitly in [Cai98b].

The smoothing parameter was defined by Micciancio and Regev [MR07]. Let us consider the Gaussian function with variance s and center $\boldsymbol{c} \in \mathbb{R}^n$ $\rho_{s,\boldsymbol{c}}(\boldsymbol{x}) =$

$\exp(-\pi \|\boldsymbol{x} - \boldsymbol{c}\|^2 / s^2)$. Let us define the Gaussian distribution $\nu_{s,\boldsymbol{c}} = \rho_{s,\boldsymbol{c}}/s^n$. (we have $\int_{\boldsymbol{x}\in\mathbb{R}^n} \nu_{s,\boldsymbol{c}}(\boldsymbol{x})d\boldsymbol{x} = 1$.) For a countable set $S \subseteq \mathbb{R}^n$, we extend the definition of $\rho_{s,\boldsymbol{c}}$ as $\rho_{s,\boldsymbol{c}}(S) = \sum_{\boldsymbol{x}\in S} \rho_{s,\boldsymbol{c}}(\boldsymbol{x})$. If $\boldsymbol{c} = \boldsymbol{0}$, we often drop $\boldsymbol{c}$ from $\rho_{s,\boldsymbol{c}}$ and $\nu_{s,\boldsymbol{c}}$.

Definition 2.1.3. For an n-dimensional lattice Λ, and positive real $\epsilon > 0$, we define its smoothing parameter $\eta_\epsilon(\Lambda)$ to be the smallest s such that $\rho_{1/s}(\Lambda^* \setminus \{\boldsymbol{0}\}) \leq \epsilon$.

As noted in [MR07], $\rho_{1/s}(\Lambda^* \setminus \{\boldsymbol{0}\})$ is a continuous and strictly decreasing function of s: $\lim_{s\to 0} \rho_{1/s}(\Lambda^* \setminus \{\boldsymbol{0}\}) = \infty$ and $\lim_{s\to\infty} \rho_{1/s}(\Lambda^* \setminus \{\boldsymbol{0}\}) = 0$. Then, $\eta_\epsilon(\Lambda)$ is also a continuous and strictly decreasing function of ϵ. This parameter was named as "smoothing," since $D_{s,\boldsymbol{c}} \bmod \mathcal{P}(\boldsymbol{B})$ is almost uniformly distributed over $\mathcal{P}(\boldsymbol{B})$. Precisely, we have the following lemma.

Lemma 2.1.4 ([MR07]). *For any $s > 0$, $\boldsymbol{c} \in \mathbb{R}^n$, and a lattice $\Lambda = \mathcal{L}(\boldsymbol{B})$, the statistical distance between $\nu_{s,\boldsymbol{c}} \bmod \mathcal{P}(\boldsymbol{B})$ and the uniform distribution over $\mathcal{P}(\boldsymbol{B})$ is at most $\frac{1}{2}\rho_{1/s}(\Lambda^* \setminus \{\boldsymbol{0}\})$. In particular, for any $\epsilon > 0$ and any $s \geq \eta_\epsilon(\Lambda)$, the statistical distance is at most*

$$\Delta(\nu_{s,\boldsymbol{c}} \bmod \mathcal{P}(\boldsymbol{B}), U(\mathcal{P}(\boldsymbol{B}))) \leq \epsilon/2.$$

For an n-dimensional lattice Λ and a lattice vector $\boldsymbol{x} \in \Lambda$, we define

$$D_{\Lambda,s,\boldsymbol{c}}(\boldsymbol{x}) = \frac{\nu_{s,\boldsymbol{c}}(\boldsymbol{x})}{\nu_{s,\boldsymbol{c}}(\Lambda)} = \frac{\rho_{s,\boldsymbol{c}}(\boldsymbol{x})}{\rho_{s,\boldsymbol{c}}(\Lambda)}.$$

These quantities relate to each other. For example, we have the following relations.

Lemma 2.1.5 ([GPV08] etc.). *For any n-dimensional lattice Λ,*

$$\lambda_1(\Lambda) \leq \widetilde{bl}(\Lambda) \leq \lambda_n(\Lambda) \leq 2\mu(\Lambda) \leq \sqrt{n} \cdot \widetilde{bl}(\Lambda).$$

The following relations with the smoothing parameter play important roles in the reductions.

Lemma 2.1.6 (Lemma 3.2 [MR07]). *For any n-dimensional lattice Λ, $\eta_\epsilon(\Lambda) \cdot \lambda_1(\Lambda^*) \leq \sqrt{n}$, where $\epsilon = 2^{-n}$.*

Lemma 2.1.7 (Lemma 3.1 [GPV08] and Lemma 3.2 [MR07]). *For any n-dimensional lattice Λ and positive real $\epsilon > 0$*

$$\eta_\epsilon(\Lambda) \leq \widetilde{bl}(\Lambda) \cdot \sqrt{\frac{1}{\pi}\ln(2n(1 + 1/\epsilon))} \leq \lambda_n(\Lambda) \cdot \sqrt{\frac{1}{\pi}\ln(2n(1 + 1/\epsilon))}.$$

In particular, for any $g(n) = \omega(\log n)$, there exists a negligible function $\epsilon(n)$ such that $\eta_\epsilon(\Lambda) \leq \sqrt{g(n)} \cdot \widetilde{bl}(\Lambda) \leq \sqrt{g(n)} \cdot \lambda_n(\Lambda)$.

Notice that, for $\epsilon \in (0, 1)$, we have that

$$\sqrt{\frac{1}{\pi}\ln(2n(1 + 1/\epsilon))} \leq \sqrt{\ln(4n\epsilon^{-1})}.$$

Hence, for any $g(n) = \omega(\log n)$, we have that

$$\sqrt{\ln(4n\epsilon^{-1})} \leq \sqrt{g(n)}$$

by setting $\epsilon(n) = 4n^{-g(n)/\ln(n)+1} = n^{-\omega(1)}$.

The following lemma clarifies the tighter relation between $\eta_\epsilon(\Lambda)$ and $\lambda_1^\infty(\Lambda^*)$.

Lemma 2.1.8 ([Pei07, Lemma 3.5] using [Ban95])**.** *For any n-dimensional lattice Λ and positive real $\epsilon > 0$*

$$\eta_\epsilon(\Lambda) \leq \frac{\sqrt{\frac{1}{\pi}\ln(2n(1+1/\epsilon))}}{\lambda_1^\infty(\Lambda^*)}.$$

In particular, for any $g(n) = \omega(\log n)$, there exists a negligible function $\epsilon(n)$ such that $\eta_\epsilon(\Lambda) \leq \sqrt{g(n)}/\lambda_1^\infty(\Lambda^*)$.

More on the smoothing parameter and Gaussian distributions: The next important property of η_ϵ is the bound on $D_{\Lambda,s,\boldsymbol{c}}$; for $s \geq \eta_\epsilon(\Lambda)$ the output has the norm at most $s\sqrt{n}$ with overwhelming probability.

Lemma 2.1.9 ([MR07, Lemma 4.4])**.** *For any n-dimensional lattice Λ, point $\boldsymbol{c} \in \mathbb{R}^n$, and reals $\epsilon \in (0,1)$ and $s \geq \eta_\epsilon(\Lambda)$,*

$$\Pr_{\boldsymbol{x} \leftarrow D_{\Lambda,s,\boldsymbol{c}}}[\|\boldsymbol{x} - \boldsymbol{c}\| > s\sqrt{n}] \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}.$$

Micciancio and Regev also bounded $\rho_{s,\boldsymbol{c}}(\Lambda)$ by $\rho_s(\Lambda)$.

Lemma 2.1.10 ([MR07] implicit in Lemma 4.4, see [GPV08])**.** *For any n-dimensional lattice Λ, point $\boldsymbol{c} \in \mathbb{R}^n$, and reals $\epsilon \in (0,1)$ and $s \geq \eta_\epsilon(\Lambda)$,*

$$\rho_{s,\boldsymbol{c}}(\Lambda) \in \left[\frac{1-\epsilon}{1+\epsilon}, 1\right] \cdot \rho_s(\Lambda).$$

We have another property of the smoothing parameter on $D_{\Lambda,s}$, which was shown in the proof of [Reg09, Lemma 3.11]. Let us consider the distribution $\boldsymbol{B}^{-1}D_{\Lambda,s} \bmod q$; (1) take a sample $\boldsymbol{y}$ from $D_{\Lambda,s}$ and (2) output $\boldsymbol{B}^{-1}\boldsymbol{y} \bmod q$. Since $\boldsymbol{y}$ is in Λ, the output lies in $\mathbb{Z}_q^n$.

Lemma 2.1.11 (Implicit in the proof of Lemma 3.11, [Reg09])**.** *For any n-dimensional lattice Λ, reals $\epsilon \in (0,1/2)$ and $s > q\eta_\epsilon(\Lambda)$,*

$$\Delta(\boldsymbol{B}^{-1}D_{\Lambda,s} \bmod q, U(\mathbb{Z}_q^n)) \leq \frac{\epsilon}{1-\epsilon}.$$

Proof. The proof is the same as the one of Regev [Reg09]. Let A be a random variable distributed according to $\boldsymbol{B}^{-1}D_{\Lambda,s} \bmod q$. Then, for any $\boldsymbol{a} \in \mathbb{Z}_q^n$,

$$\Pr_A[A = \boldsymbol{a}] = \frac{\rho_s(q\Lambda + \Lambda\boldsymbol{a})}{\sum_{\boldsymbol{b} \in \mathbb{Z}_q^n} \rho_s(q\Lambda + \Lambda\boldsymbol{b})}.$$

Suppose that we take s sufficiently large satisfying $\eta_\epsilon(\Lambda) < s/q$, that is, $\eta_\epsilon(q\Lambda) < s$. By the claim below in [Reg09], we have that

$$\rho_s(q\Lambda + \Lambda\boldsymbol{a}) \in (1 \pm \epsilon)s^n \det((q\Lambda)^*) = (1 \pm \epsilon)(s/q)^n \det(\Lambda^*).$$

Hence,

$$\Pr_A[A = \boldsymbol{a}] \in \frac{(1 \pm \epsilon)(s/q)^n \det(\Lambda^*)}{\sum_{\boldsymbol{b}}(1 \mp \epsilon)(s/q)^n \det(\Lambda^*)} = \left[\frac{1-\epsilon}{1+\epsilon}, \frac{1+\epsilon}{1-\epsilon}\right] \cdot q^{-n}.$$

In addition, we have $\left|1 - (\frac{1+\epsilon}{1-\epsilon})\right| = \frac{2\epsilon}{1-\epsilon}$ and $\left|1 - (\frac{1-\epsilon}{1+\epsilon})\right| = \frac{2\epsilon}{1+\epsilon}$. Therefore, the statistical distance is at most

$$\Delta(A, U(\mathbb{Z}_q^n)) \leq \frac{\epsilon}{1-\epsilon}$$

and this completes the proof.

Claim 2.1.12 (Claim 3.8, [Reg09])**.** *For any lattice Λ, point $\boldsymbol{c} \in \mathbb{R}^n$, and any reals $\epsilon > 0$ and $s \geq \eta_\epsilon(\Lambda)$,*

$$(1-\epsilon)s^n \det(\Lambda^*) \leq \rho_s(\Lambda + \boldsymbol{c}) \leq (1+\epsilon)s^n \det(\Lambda^*).$$

□

By the similar argument, we can show the following generalized lemma.

Lemma 2.1.13 (Corollary 2.8, [GPV08])**.** *Let Λ and Λ' be n-dimensional lattices with $\Lambda' \subseteq \Lambda$. Then for any $\epsilon \in (0, 1/2)$, any $s \geq \eta_\epsilon(\Lambda')$, and any $\boldsymbol{c} \in \mathbb{R}^n$,*

$$\Delta(D_{\Lambda,s,\boldsymbol{c}} \bmod \Lambda', U(\Lambda \bmod \Lambda')) \leq 2\epsilon.$$

The final lemma ensures the min-entropy of $D_{\Lambda,s,\boldsymbol{c}}$.

Lemma 2.1.14 ([PR06])**.** *For any n-dimensional lattice Λ, point $\boldsymbol{c} \in \mathbb{R}^n$, reals $\epsilon > 0$ and $s \geq 2\eta_\epsilon(\Lambda)$, and for every $\boldsymbol{x} \in \Lambda$,*

$$D_{\Lambda,s,\boldsymbol{c}}(\boldsymbol{x}) \leq \frac{1+\epsilon}{1-\epsilon} \cdot 2^{-n}.$$

In particular, for $\epsilon < 1/3$, the min-entropy of $D_{\Lambda,s,\boldsymbol{c}}$ is at least $n-1$.

2.2 Lattice Problems

We give the definitions of well-known lattice problems, the Shortest Vector Problem (SVP^p) and its approximation version (SVP$_\gamma^p$):

Definition 2.2.1 (Shortest Vector Problem, SVP)**.** The problem SVP^p is, given a basis $\boldsymbol{B}$ of a lattice Λ, finding the shortest non-zero vector $\boldsymbol{v}$ in Λ in the l_p norm.

Definition 2.2.2 (Approximation version of SVP)**.** The problem SVP$_\gamma^p$ is, given a basis $\boldsymbol{B}$ of a lattice Λ, finding a non-zero vector $\boldsymbol{v}$ in Λ such that for any non-zero vector $\boldsymbol{x}$ in Λ, $\|\boldsymbol{v}\|_p \leq \gamma \|\boldsymbol{x}\|_p$.

We next give the definition of the gap version of $\mathrm{SVP}_{\gamma}^{p}$.

Definition 2.2.3 (Gap version of SVP, GapSVP)**.** For a gap function γ, an instance of $\mathrm{GapSVP}_{\gamma}^{p}$ is a pair $(\boldsymbol{B}, d)$ where $\boldsymbol{B}$ is a basis of a lattice Λ and d is a rational number. In YES input there exists a vector $\boldsymbol{v} \in \Lambda \setminus \{\boldsymbol{0}\}$ such that $\|\boldsymbol{v}\|_p \le d$, that is, $\lambda_1^p(\Lambda) \le d$. In NO input, for any vector $\boldsymbol{v} \in \Lambda \setminus \{\boldsymbol{0}\}$, $\|\boldsymbol{v}\|_p > \gamma d$, that is, $\lambda_1^p(\Lambda) > \gamma d$.

Apparently, the smaller γ, the harder the problems, SVP_{γ} and GapSVP_{γ}, are.

Peikert also define the variant of GapSVP, ζ-to-γ-GapSVP in which we are given a lattice having a vector shorter than $\zeta(n)$.

Definition 2.2.4 ($\mathrm{GapSVP}_{\zeta,\gamma}$)**.** For functions $\zeta(n) \ge \gamma(n) \ge 1$, an input to ζ-to-γ-GapSVP, $\mathrm{GapSVP}_{\zeta,\gamma}$, is a pair $(\boldsymbol{B}, d)$, where:

- $\boldsymbol{B}$ is a basis of an n-dimensional lattice Λ for which $\lambda_1(\Lambda) \le \zeta(n)$,
- $\min_i \|\tilde{\boldsymbol{b}}_i\| \ge 1$, and
- $1 \le d \le \zeta(n)/\gamma(n)$.

It is a YES instance if $\lambda_1(\Lambda) \le d$, and is a NO instance if $\lambda_1(\Lambda) > \gamma(n) \cdot d$.

Note that, for any $\zeta(n) \ge 2^{n/2}$, $\mathrm{GapSVP}_{\zeta,\gamma}$ is at least hard as GapSVP_{γ}, since we can reduce the basis so that $\lambda_1(\Lambda) \le \|\boldsymbol{b}_i\| \le 2^{n/2} \cdot \min_i \|\tilde{\boldsymbol{b}}_i\|$ by using the LLL algorithm [LLL82].

The shortest independent vectors problem gives also the base of the cryptographic scheme.

Definition 2.2.5 (Shortest Independent Vectors Problem, SIVP)**.** The problem SIVP^p is, given a basis $\boldsymbol{B}$ of a lattice Λ, finding the shortest independent vectors $\boldsymbol{S}$ in Λ in the l_p norm. That is, finding $\boldsymbol{S}$ such that $\|\boldsymbol{S}\|_p = \lambda_n^p(\Lambda)$.

Definition 2.2.6 (Approximation version of SIVP)**.** The problem $\mathrm{SVP}_{\gamma}^{p}$ is, given a basis $\boldsymbol{B}$ of a lattice Λ, finding independent vectors $\boldsymbol{S}$ such that $\|\boldsymbol{S}\|_p \le \gamma \cdot \lambda_n^p(\Lambda)$.

The above definition is generalized with some lattice constant ϕ.

Definition 2.2.7 (Generalized Independent Vectors Problem, GIVP)**.** The problem $\mathrm{GIVP}_{\gamma}^{\phi,p}$ is, given a basis $\boldsymbol{B}$ of a lattice Λ, finding n linearly independent vectors $\boldsymbol{S} \subset \Lambda$ such that $\|\boldsymbol{S}\|_p \le \gamma \cdot \phi(\Lambda)$.

The Closest Vector Problem (CVP^p) also often appeared in the lattice-based cryptography. We give the definitions of CVP^p, the approximation version $\mathrm{CVP}_{\gamma}^{p}$, and the gap version $\mathrm{GapCVP}_{\gamma}^{p}$.

Definition 2.2.8 (Closest Vector Problem, CVP)**.** The problem CVP^p is, given a basis $\boldsymbol{B}$ of a lattice Λ and a target vector $\boldsymbol{t}$, finding the closest vector $\boldsymbol{v}$ in Λ to $\boldsymbol{t}$ in the l_p norm, that is, finding a vector $\boldsymbol{v}$ such that for any $\boldsymbol{x} \in \Lambda$, $\|\boldsymbol{v} - \boldsymbol{t}\|_p \le \|\boldsymbol{x} - \boldsymbol{t}\|$.

Definition 2.2.9 (Approximation version of CVP)**.** The problem $\mathrm{CVP}_{\gamma}^{p}$ is, given a basis $\boldsymbol{B}$ of a lattice Λ and a target vector $\boldsymbol{t}$, finding a vector $\boldsymbol{v}$ in Λ such that for any vector $\boldsymbol{x}$ in Λ, $\|\boldsymbol{v} - \boldsymbol{t}\|_p \le \gamma \|\boldsymbol{x} - \boldsymbol{t}\|_p$.

Definition 2.2.10 (Gap version of CVP, GapCVP)**.** For a gap function γ, an instance of GapCVP_γ^p is a triplet $(\boldsymbol{B}, \boldsymbol{t}, d)$ where $\boldsymbol{B}$ is a basis of a lattice Λ, $\boldsymbol{t}$ is a target vector in $\mathbb{Q}^m$, and d is a rational number. In YES input there exists a vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq d$. In NO input, for any vector $\boldsymbol{v} \in \Lambda$, $\|\boldsymbol{v} - \boldsymbol{t}\|_p > \gamma d$.

In addition, we give the definition of Bounded Distance Decoding problem, a promise version of CVP.

Definition 2.2.11 (Bounded Distance Decoding, BDD)**.** The problem BDD is, given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ, a real $d > 0$, and a target point $\boldsymbol{t} \in \mathbb{R}^n$ such that $\text{dist}(\boldsymbol{t}, \Lambda) \leq d$, finding the close lattice vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v} - \boldsymbol{t}\| \leq d$.

We can give a generalized version of bounded distance decoding as follows:

Definition 2.2.12 (Guaranteed Distance Decoding $\text{GDD}_\gamma^{\phi,p}$)**.** The problem $\text{GDD}_\gamma^{\phi,p}$ is, given a basis $\boldsymbol{B}$ of a lattice Λ and a target point $\boldsymbol{t} \in \mathbb{R}^n$, finding a lattice vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \gamma\phi(\Lambda)$.

Computing covering radius for any lattice is also hard problem. We give the definitions of the covering radius problem and its gap version.

Definition 2.2.13 (Covering Radius Problem, CRP)**.** The problem CRP^p is, given a basis $\boldsymbol{B}$ of a lattice Λ, finding the covering radius $\mu(\Lambda)$ of the lattice Λ.

Definition 2.2.14 (Gap version of Covering Radius Problem, GapCRP_γ)**.** For a gap function γ, an instance of GapCRP_γ^p is a pair $(\boldsymbol{B}, d)$ where $\boldsymbol{B}$ is a basis of a lattice Λ and d is a rational number. In YES inputs, $\mu(\Lambda) \leq d$ and in NO inputs, $\mu(\Lambda) > \gamma \cdot d$.

Micciancio and Regev defined a new problem, incremental guaranteed distance decoding problem, IncGDD which is the variant of IncSIVP in Ajtai [Ajt96].

Definition 2.2.15 (Incremental Guaranteed Distance Decoding, IncGDD [MR07])**.** An input to $\text{IncGDD}_{\gamma,g}^{p,\phi}$ is a quadruplet $(\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{t}, r)$, where $\boldsymbol{B}$ is a basis for a full-rank lattice Λ in $\mathbb{R}^n$, $\boldsymbol{S} \subset \Lambda$ is a full-rank set of lattice vectors, $\boldsymbol{t} \in \mathbb{R}^n$ is a target point, and r is a real with $r > \gamma \cdot \phi(\Lambda)$. The problem is finding a lattice vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \frac{1}{g} \|\boldsymbol{S}\|_p + r$.

Gentry, Peikert, and Vaikuntanathan [GPV08] pointed out that a slightly simpler problem suffices for the reductions in Micciancio and Regev [MR07]. The problem is incremental independent vectors decoding problem IncIVD defined as follows:

Definition 2.2.16 (Incremental Independent Vectors Decoding, IncIVD [GPV08])**.** An input to $\text{IncIVD}_{\gamma,g}^{p,\phi}$ is a triplet $(\boldsymbol{B}, \boldsymbol{S}, \boldsymbol{t})$, where $\boldsymbol{B}$ is a basis for a full-rank lattice Λ in $\mathbb{R}^n$, $\boldsymbol{S} \subset \Lambda$ is a full-rank set of lattice vectors such that $\|\boldsymbol{S}\|_p \geq \gamma \cdot \phi(\boldsymbol{B})$, and $\boldsymbol{t} \in \mathbb{R}^n$ is a target point. The problem is finding a lattice vector $\boldsymbol{v} \in \Lambda$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \|\boldsymbol{S}\|_p / g$.

Finally, we review the discrete Gaussian Sampling problem (DGS) in [Reg09].

Definition 2.2.17 (Discrete Gaussian Sampling, DGS [Reg09])**.** The problem DGS_ϕ is, given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ and a real $s > \phi(\Lambda)$, sampling from $D_{\Lambda,s}$.

Reductions from lattice problems to IncIVD

The basic reductions appeared in the textbook of Micciancio and Goldwasser [MG02]. We here show reductions from several lattice problems, GIVP, GDD, GapCRP, and GapSVP to IncIVD. These problems will be the underlying problems of several cryptographic primitives in this thesis through IncIVD.

Micciancio [Mic07] and Micciancio and Regev [MR07] showed these reductions to IncGDD. Gentry et al. [GPV08] improved the average-case/worst-case reductions and noted that there are reductions from GIVP, GDD, GapCRP to IncIVD, but they omitted the proofs. For completeness, we prove them by modifying the proofs (or the proof sketches) in [Mic07, MR07].

Lemma 2.2.18. *There is a lattice-preserving polynomial-time reduction from* $\mathrm{GIVP}^{\phi,p}_{4\gamma}$ *to* $\mathrm{IncIVD}^{\phi,p}_{\gamma,4}$.

Proof. The proof is an adapted version of the one in [Mic07, Lemma 4.6]. We define the reduction algorithm as follows:

1. Scan $i \in [n]$ such that $\|\boldsymbol{s}_i\|_p = \|\boldsymbol{S}\|_p$.
2. Let $\boldsymbol{t}$ be an orthogonal vector of length $\|\boldsymbol{S}\|_p/2$ to the hyperplane $\mathrm{span}(\boldsymbol{s}_1,\dots,\boldsymbol{s}_{i-1},\boldsymbol{s}_{i+1},\dots,\boldsymbol{s}_n)$.
3. Invoke the oracle $\mathcal{O}$ on $(\boldsymbol{B},\boldsymbol{S},\boldsymbol{t})$ and obtain $\boldsymbol{v} \in \Lambda$.
4. If $\mathcal{O}$ fails output $\boldsymbol{S}$.
5. Replace $\boldsymbol{S}$ with $\boldsymbol{S}' = [\boldsymbol{s}_1,\dots,\boldsymbol{s}_{i-1},\boldsymbol{v},\boldsymbol{s}_{i+1},\dots,\boldsymbol{s}_n]$ and go to Step 1.

Suppose that $\boldsymbol{v}$ is a valid solution of $\mathrm{IncIVD}^{\phi,p}_{\gamma,4}$ on input $(\boldsymbol{B},\boldsymbol{S},\boldsymbol{t})$. Then, we have that $\|\boldsymbol{v}-\boldsymbol{t}\|_p \le \|\boldsymbol{S}\|_p/4$ and, hence, $\boldsymbol{v}$ is not included in the hyperplane spanned by $\boldsymbol{s}_1,\dots,\boldsymbol{s}_{i-1},\boldsymbol{s}_{i+1},\dots,\boldsymbol{s}_n$, which shows the linear independence of $\boldsymbol{S}' \subset \Lambda$. We also have that $\|\boldsymbol{v}\|_p \le \|\boldsymbol{v}-\boldsymbol{t}\|_p + \|\boldsymbol{t}\|_p \le \|\boldsymbol{S}\|_p/4 + \|\boldsymbol{S}\|/2 = \frac{3}{4}\|\boldsymbol{S}\|_p$.

Hence, by repeating the above procedure until the oracle $\mathcal{O}$ fails, that is, $\|\boldsymbol{S}\|_p/4 \le \gamma\phi(\Lambda)$, and we obtain the short linearly independent vectors $\boldsymbol{S} \subset \Lambda$ with $\|\boldsymbol{S}\|_p \le 4\gamma \cdot \phi(\Lambda)$. □

Lemma 2.2.19. *There is a lattice-preserving polynomial-time reduction from* $\mathrm{GDD}^{\phi}_{2\gamma}$ *to* $\mathrm{IncIVD}^{\phi}_{\gamma,4}$.

Proof. As in the proof in [Mic07, Lemma 4.7], by applying the reduction algorithm in the previous proof, we obtain a full-rank sets $\boldsymbol{S} \subset \Lambda$ such that $\|\boldsymbol{S}\|_p \le 4\gamma \cdot \phi(\Lambda)$. Since Micciancio's proof for the reduction to IncGDD exploited r, we give another reduction algorithm for a reduction to IncIVD, which exploits $\boldsymbol{S}$.

After obtaining $\boldsymbol{S}$, we then run the following algorithm:

1. Scan $i \in [m]$ such that $\|\boldsymbol{s}_i\|_p = \|\boldsymbol{S}\|_p$ and set $j \leftarrow 1$.

2. Repeat the following procedure.
   (a) $\boldsymbol{S}^{(j)} \leftarrow [\boldsymbol{s}_1, \dots, \boldsymbol{s}_{i-1}, 2^j \boldsymbol{s}_i, \boldsymbol{s}_{i+1}, \dots, \boldsymbol{s}_n]$.
   (b) Invoke the oracle O on input $(\boldsymbol{B}, \boldsymbol{S}^{(j)}, \boldsymbol{t})$ and obtain $\boldsymbol{v}$.
   (c) If $\boldsymbol{v} \in \Lambda$ and $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \|\boldsymbol{S}^{(j)}\|_p/4$ then output $\boldsymbol{v}$.
   (d) Otherwise, increment j and go to step (a).

Notice that $\boldsymbol{S}^{(j)}$ is full-rank (since $\det(\boldsymbol{S}^{(j)}) = 2^j \det(\boldsymbol{S}) \neq 0$) and $\|\boldsymbol{S}^{(j)}\|_p = 2\|\boldsymbol{S}^{(j-1)}\|_p$. We now consider the final step j. Then, the check must fail in the $(j-1)$-th repeat and we have that $\|\boldsymbol{S}^{(j-1)}\|_p \leq 4\gamma \cdot \phi(\Lambda)$. Thus, $\|\boldsymbol{S}^{(j)}\|_p \leq 8\gamma \cdot \phi(\Lambda)$ and we can upper bound $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \|\boldsymbol{S}^{(j)}\|_p/4 \leq 2\gamma \cdot \phi(\Lambda)$, as required. □

Lemma 2.2.20 ([MR07, Lemma 5.12]). *For any $\gamma = \gamma(n)$, there exists a lattice-preserving randomized reduction from* GapCRP$_\gamma$ *to* GDD$^{\lambda_n}_{\gamma/4}$. *In particular, there is a lattice-preserving randomized reduction from* GapCRP$_{8\gamma}$ *to* IncIVD$^{\lambda_n}_{\gamma,4}$.

The proof is in [MR07].

2.3 Hardness of Lattice Problems

We discuss the hardness results on lattice problems.

The NP-hardness of CVP^p for any p was shown by van Emde Boas [vEB81]. Arora, Babai, Stern, and Sweedyk [ABSS97] showed the NP-hardness of CVP^p_c for any constant c. Dinur, Kindler, Raz, and Safra [DKRS03] improved the approximation factor to $2^{O(\log n/\log\log n)} = n^{1/\log\log n}$. On CVP$_\gamma$, the major problem is showing NP-hardness for approximation factor n^ϵ for small constant $\epsilon > 0$.

The first result of the NP-hardness of SVP is van Emde Boas [vEB81] which showed for the l_∞ norm. Later, Ajtai [Ajt98] showed that SVP$_\gamma$ is NP-hard under randomized reductions for $\gamma = 1 + 2^{-cn}$ for some constant c. Cai and Nerurkar [CN97] improved the approximation factor $1 + 1/n^\epsilon$ for any fixed $\epsilon > 0$. Micciancio gave the proof for approximation factor $\sqrt{2}$ under RUR-reductions in [Mic00]. Khot [Kho06] showed that, assuming NP $\not\subseteq$ ZPP, SVP$^p_\gamma$ for $\gamma = p^{1-\epsilon}$ is intractable for all integers $p \geq p(\epsilon)$. Khot [Kho05] proved that SVP$_c$ is NP-hard under the assumption NP $\not\subseteq$ RP for any constant c. He also proved that SVP$_\gamma$ for $\gamma = 2^{O((\log n)^{1/2-\epsilon})}$ is NP-hard within under the assumption NP $\not\subseteq$ RTIME$(2^{\mathrm{poly}(\log n)})$. Haviv and Regev [HR07] improved the approximation factor to $\gamma = 2^{(\log n)^{1-\epsilon}}$ for any ϵ under the same assumption.

Even within a polynomial approximation factor, it is unknown whether there exists a polynomial-time algorithm for the approximation version of SVP. The most well-known solution to this approximation problem is the so-called LLL algorithm proposed in [LLL82]. This algorithm can solve SVP$_{2^{n/2}}$ in polynomial time. Schnorr [Sch87] generalized the LLL algorithm which solves SVP$_\gamma$ for $\gamma = (1+c)^n$ for any constant $c > 0$.

There are several exponential-time algorithms for SVP and CVP. For the old results, see the survey [AEVZ02]. Ajtai, Kumar, and Sivakumar [AKS01] proposed the randomized algorithm for SVP which runs in exponential time of the dimension. Nguyen and Vidick [NV08] implemented this and clarified the time and space complexity of this algorithm, the time is $\tilde{O}(2^{5.9n})$ and the space is $\tilde{O}(2^{2.95n})$, which improved the analysis by Regev [Reg04a, Lecture 8]. This is improved by Micciancio and Voulgaris [MV09], whose probabilistic algorithm runs in time $2^{3.199n}$ and space $2^{1.325n}$. (Very recently, they also proposed a *deterministic* algorithm running within time $2^{O(n)}$ [MV10], which will be verified by peer reviews.)

On the other hand, there are several non-NP-hardness results on the approximation version of SVP with a polynomial approximation factor. Goldreich and Goldwasser [GG00] showed $\mathrm{SVP}_{\Omega(\sqrt{n/\log n})}$ is in NP $\cap$ coAM. Aharonov and Regev [AR05] showed that $\mathrm{SVP}_{\Omega(\sqrt{n})}$ is in NP $\cap$ coNP.

The unique shortest vector problem (uSVP) is also well known as a hard lattice problem applicable to cryptographic constructions. We say the shortest vector $\boldsymbol{v}$ of a lattice Λ is f-unique if for any non-zero vector $\boldsymbol{x} \in \Lambda$ which is not parallel to $\boldsymbol{v}$, $f\,\|\boldsymbol{v}\| \leq \|\boldsymbol{x}\|$. The definition of uSVP is given as follows.

Definition 2.3.1 (f-uSVP)**.** Given a basis $\boldsymbol{B}$ of a lattice Λ whose shortest vector is f-unique, find a non-zero vector $\boldsymbol{v} \in \Lambda$ such that for any non-zero vector $\boldsymbol{x} \in \Lambda$ which is not parallel to $\boldsymbol{v}$, $f\,\|\boldsymbol{v}\| \leq \|\boldsymbol{x}\|$.

Similarly to the case of SVP, the exact version of uSVP is shown to be in NP-hard by Kumar and Sivakumar [KS01]. Cai [Cai98b] showed that $\Omega(n^{1/4})$-uSVP is in NP $\cap$ coAM.

In addition, recent results by Lyubashevsky and Micciancio [LM09] indicates the relations on GapSVP, BDD, and uSVP. They showed, up to a small polynomial factor $\sqrt{n/\log n}$, the equivalence of the uSVP, BDD, and GapSVP; $\mathrm{GapSVP}_\gamma \geq \mathrm{uSVP}_\gamma$ for any $\gamma \geq 1$, $\mathrm{BDD}_{\frac{1}{\gamma}\sqrt{n/\log n}} \geq \mathrm{GapSVP}_\gamma$ for $\gamma > 2\sqrt{n/\log n}$, and $\mathrm{uSVP}_\gamma \geq \mathrm{BDD}_{1/(2\gamma)}$ for $\gamma \geq 1$.

2.4 Average-Case/Worst-Case Reductions

Before giving the reviews of the reductions, we first review lattices, *q-ary lattices*, which are relevant to linear codes.

2.4.1 Linear Codes and q-Ary Lattices

Linear Codes: We start with the definition of codes and linear codes. Let Σ be a finite alphabet of size q and let m be a block length. Then a code is a subset of Σ^m. Let $\mathbb{F} = \mathbb{F}_q$ be a field of cardinality q. Then a linear code C is a *subspace* of $\mathbb{F}_q^m$. The dimension of the code C is naturally defined.

We say $\boldsymbol{G}$ as a generator matrix of C if $C = \{\boldsymbol{G}^T\boldsymbol{s} \in \mathbb{F}^m \mid \boldsymbol{s} \in \mathbb{F}^n\}$. We say $\boldsymbol{H}$ as a parity-check matrix of C if $C = \{\boldsymbol{e} \in \mathbb{F}^m \mid \boldsymbol{He} = \boldsymbol{0} \in \mathbb{F}^n\}$.

For a matrix $\boldsymbol{A} \in \mathbb{F}^{n\times m}$, a code having a generator matrix $\boldsymbol{A}$ is denoted by $C_G(\boldsymbol{A})$, that is, $\{\boldsymbol{A}^T\boldsymbol{s} \in \mathbb{F}^m \mid \boldsymbol{s} \in \mathbb{F}^n\}$. A code having a parity-check matrix $\boldsymbol{A}$ is denoted by $C_H(\boldsymbol{A})$, that is, $\{\boldsymbol{e} \in \mathbb{F}^m \mid \boldsymbol{Ae} = \boldsymbol{0}\}$.

***q*-ary lattices:** For a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, we define two sublattices of $\mathbb{Z}^m$,

$$\Lambda_q(\boldsymbol{A}) = \{\boldsymbol{p} \in \mathbb{Z}^m \mid \exists \boldsymbol{s}, \boldsymbol{A}^T\boldsymbol{s} \equiv \boldsymbol{p} \pmod{q}\},$$
$$\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{e} \in \mathbb{Z}^m \mid \boldsymbol{Ae} \equiv \boldsymbol{0} \pmod{q}\}.$$

It is obvious that two sets are lattices because they are discrete and additive subgroup of $\mathbb{Z}^m$. It is also obvious that $q\boldsymbol{I} \subset \Lambda_q(\boldsymbol{A}), \Lambda_q^{\perp}(\boldsymbol{A})$. Hence, they are superlattices of $q\mathbb{Z}^m$ and thus full-rank.

In addition, you can confirm that they are relevant to linear codes. The former lattice $\Lambda_q(\boldsymbol{A})$ is $q\mathbb{Z}+C_G(\boldsymbol{A})$, where + denotes the Minkowski sum. The latter lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ is also $q\mathbb{Z} + C_H(\boldsymbol{A})$. By a simple calculation, we confirm that $(\Lambda_q(\boldsymbol{A}))^* = \frac{1}{q}\Lambda_q^{\perp}(\boldsymbol{A})$.

Lemma 2.4.1. *For any matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$*,* $(\Lambda_q(\boldsymbol{A}))^* = \frac{1}{q}\Lambda_q^{\perp}(\boldsymbol{A})$.

Proof. ($\supseteq$) Consider any vector $\boldsymbol{e} \in \Lambda_q^{\perp}(\boldsymbol{A})$. We show that, for any vector $\boldsymbol{y} \in \Lambda_q(\boldsymbol{A})^*$, $\langle \frac{1}{q}\boldsymbol{e}, \boldsymbol{y}\rangle \in \mathbb{Z}$. Since $\boldsymbol{y}$ is in $\Lambda_q(\boldsymbol{A})$, there is some vector $\boldsymbol{s} \in \mathbb{Z}^n$ such that $\boldsymbol{y} \equiv \boldsymbol{A}^T\boldsymbol{s} \pmod{q}$. Hence,

$$\langle \boldsymbol{e}, \boldsymbol{y}\rangle \equiv \boldsymbol{e}^T\boldsymbol{A}^T\boldsymbol{s} \equiv \boldsymbol{0}^T\boldsymbol{s} \equiv 0 \pmod{q}.$$

This shows that $\langle \frac{1}{q}\boldsymbol{e}, \boldsymbol{y}\rangle \in \mathbb{Z}$ and $(\Lambda_q(\boldsymbol{A}))^* \supseteq \frac{1}{q}\Lambda_q^{\perp}(\boldsymbol{A})$.

($\subseteq$) Instead of the statement, we show the tautological statement $q(\Lambda_q(\boldsymbol{A}))^* \subseteq \Lambda_q^{\perp}(\boldsymbol{A})$. Consider any vector $\boldsymbol{x}$ in $(\Lambda_q(\boldsymbol{A}))^*$ and suppose that $q\boldsymbol{x}$ is *not* in $\Lambda_q^{\perp}(\boldsymbol{A})$. By this hypothesis, we have that $q\boldsymbol{Ax} \not\equiv \boldsymbol{0} \pmod{q}$, which indicates $\boldsymbol{Ax} \notin \mathbb{Z}^n$. However, the transposes of the rows of $\boldsymbol{A}$ is in $\Lambda_q(\boldsymbol{A})$ and $\boldsymbol{Ax}$ must be in $\mathbb{Z}^n$. This means a contradiction and we complete the proof. □

The reason of why we need these lattices is clarified in the follow-on sections.

2.4.2 From the Small Integer Solution Problem

We define the Small Integer Solution problem SIS (in the l_p norm), which is often considered in the context of average-case/worst-case connections and a source of lattice-based hash functions as we see later.

Definition 2.4.2 ($\mathrm{SIS}_{q,m,\beta}^p$ [MR07])**.** For a fixed integer q and a real β, given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, the problem is finding a non-zero integer vector $\boldsymbol{e} \in \mathbb{Z}^m$ such that $\boldsymbol{Ae} \equiv \boldsymbol{0} \pmod{q}$ and $\|\boldsymbol{e}\|_p \le \beta$.

Definition 2.4.3 ($\mathrm{ISIS}_{q,m,\beta}^p$ [MR07])**.** For a fixed integer q and a real β, given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{u} \in \mathbb{Z}_q^n$, the problem is finding an integer vector $\boldsymbol{e} \in \mathbb{Z}^m$ such that $\boldsymbol{Ae} \equiv \boldsymbol{u} \pmod{q}$ and $\|\boldsymbol{e}\|_p \le \beta$.

The former problem $\mathrm{SIS}^{p}_{q,m,\beta}$ is indeed lattice problem for the q-ary lattices: Given a matrix $\boldsymbol{A}$, find the short non-zero vector $\boldsymbol{e}$ in the lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\boldsymbol{e}\|_p \leq \beta$. This problem is firstly appeared with no explicit name in the seminal paper of Ajtai [Ajt96]. The latter problem $\mathrm{ISIS}^{p}_{q,m,\beta}$ is also lattice problem for the q-ary lattices similar to CVP; Consider the lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ and find a vector $\boldsymbol{t} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{t} \equiv \boldsymbol{u} \pmod{q}$ by using the linear algebra. Then, find the lattice vector $\boldsymbol{v} \in \Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\boldsymbol{v} - \boldsymbol{t}\|_p \leq \beta$. Finally, set $\boldsymbol{e} = \boldsymbol{v} - \boldsymbol{t}$.

In this thesis, we review the average-case/worst-case reductions to SIS, which is initiated by Ajtai [Ajt96] and followed several improvements [GGH96, CN97, Mic04, MR07, GPV08], especially, the reduction by Gentry, Peikert, and Vaikuntanathan [GPV08].

Ajtai originally proposed the reduction from IncSIVP to SIS, Cai and Nerurkar [CN97] and Micciancio [Mic04] followed this. This is simplified by Micciancio and Regev introducing the new problem IncGDD. Finally, Gentry et al. improved it by showing the reduction from IncIVD. In this thesis, we use the intermediate problem IncIVD, instead of IncSIVP and IncGDD to reduce the discussions.

Gentry et al.'s reduction

Using the direct sampling of lattice points, they make a simpler reduction. We quickly introduce a sampler algorithm SampleD which, given a basis $\boldsymbol{T}$ of a lattice Λ such that $\|\tilde{\boldsymbol{T}}\| \leq L$, a real $s > L \cdot \omega(\sqrt{\log n})$, and a center $\boldsymbol{c}$, samples a vector $\boldsymbol{y}$ on a lattice Λ. The distribution of output is within a negligible distance from $D_{\Lambda,s,\boldsymbol{c}}$. For the details of SampleD, see Section 10.4. To simplify notation, we consider that we can directly sample $\boldsymbol{y}$ from $D_{\Lambda,s,\boldsymbol{c}}$ instead of using SampleD, which introduces negligible errors.

The reduction algorithm of Gentry et al. is as follows:

1. (*Setup.*) Choose an index $j \leftarrow [m]$ and $\alpha \leftarrow \{-\beta, \dots, -1, 1, \dots, \beta\}$ uniformly at random. Let $\boldsymbol{c}_j = \frac{q}{\alpha}\boldsymbol{t} \in \mathbb{R}^n$ and let $\boldsymbol{c}_i = \boldsymbol{0} \in \mathbb{R}^n$ other $i \in [m] \setminus \{j\}$. For reducing to ISIS, choose $\boldsymbol{u} \leftarrow \mathbb{Z}_q^n$ uniformly at random. For reducing to SIS, set $\boldsymbol{u} = \boldsymbol{0}$. Let $\boldsymbol{x}_j = \alpha^{-1}\boldsymbol{u} \bmod q$ and $\boldsymbol{x}_i = \boldsymbol{0}$ for $i \in [m] \setminus \{j\}$. Define the matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_m] \in \mathbb{Z}_q^{n \times m}$. Using MGReduce, obtain a basis $\boldsymbol{T}$ of $\Lambda(\boldsymbol{B})$ such that $\|\tilde{\boldsymbol{T}}\| \leq \|\tilde{\boldsymbol{S}}\| \leq \|\boldsymbol{S}\|$.
2. (*Sampling.*) Let $s = \frac{q}{\gamma}\|\boldsymbol{S}\|$. For each $i \in [m]$, sample $\boldsymbol{y}_i \leftarrow D_{\Lambda(\boldsymbol{B}),s,\boldsymbol{c}_i}$. Define the matrix $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_m] \in \mathbb{R}^{n \times m}$. Define $\boldsymbol{A} = (\boldsymbol{B}^{-1}\boldsymbol{Y} + \boldsymbol{X}) \bmod q$.
3. (*Invoking and Combining.*) Invoke the oracle $\mathcal{O}$ on $(q, \boldsymbol{A}, \boldsymbol{u}, \beta)$ and obtain $\boldsymbol{e} \in \mathbb{Z}^m$. Output the vector $\boldsymbol{v} = \frac{1}{q}\boldsymbol{Y}\boldsymbol{e}$.

Theorem 2.4.4 ([GPV08])**.** *Let $m = m(n)$, $q = q(n)$, $\beta = \beta(n)$, $\gamma = \gamma(n)$ be polynomially-bounded functions. For any $q \geq \gamma \cdot \omega(\sqrt{\log n})$, The above reduction is a probabilistic polynomial-time reduction from solving* $\mathrm{IncIVD}^{\eta_\epsilon}_{\gamma,g}$ *for $\gamma = g\beta\sqrt{n}$ in* *the worst case to solving either* $\mathrm{SIS}_{q,m,\beta}$ *or* $\mathrm{ISIS}_{q,m,\beta}$ *on average with non-negligible probability.*

In a typical case we often set $m = O(n \log q)$, $\beta = \sqrt{m}$, and g constant and obtain $\gamma = O(n\sqrt{\log q})$ and $q = \tilde{O}(n)$.

Combining the above with Lemma 2.2.18 and Lemma 2.1.7, the following corollary holds.

Corollary 2.4.5 (Implicit in [GPV08])**.** *Let m, q, β, γ be as in the above. There is a probabilistic polynomial-time reduction from* $\mathrm{GIVP}_{4\gamma}^{\eta_\epsilon}$ *in the worst case to* $\mathrm{SIS}_{q,m,\beta}$ *or* $\mathrm{ISIS}_{q,m,\beta}$ *on average.*

In particular, let $\epsilon = \epsilon(n)$ be some negligible function in n. Then, we have a probabilistic polynomial-time reduction from $\mathrm{SIVP}_{\gamma'}$ *in the worst case to* $\mathrm{SIS}_{q,m,\beta}$ *or* $\mathrm{ISIS}_{q,m,\beta}$ *on average, where $\gamma' = \gamma(n) \cdot \omega(\sqrt{\log n}) = 4\beta\sqrt{n} \cdot \omega(\sqrt{\log n})$.*

Proof: We include the proof of theorem to consistency. The following sequence of claims show the correctness of the reduction.

Claim 2.4.6. *For any values j and α, the distribution of $\boldsymbol{A}$ is statistically close to uniform over $\mathbb{Z}_q^{n\times m}$. In particular, O outputs a non-zero solution $\boldsymbol{e} \in \mathbb{Z}^m$ such that $e_j = \alpha$ with non-negligible probability.*

Proof. We have $s = \|\boldsymbol{S}\| q/\gamma \geq \|\tilde{\boldsymbol{T}}\| \cdot \omega(\sqrt{\log n})$ and the output of the sampling algorithm SampleD is distributed within negligible distance from $D_{\mathcal{L}(\boldsymbol{B}),s,\boldsymbol{c}_i}$.

We also have $\|\boldsymbol{S}\| \geq \gamma \cdot \eta_\epsilon(\mathcal{L}(\boldsymbol{B}))$, so $s \geq q \cdot \eta_\epsilon(\mathcal{L}(\boldsymbol{B})) = \eta_\epsilon(q\mathcal{L}(\boldsymbol{B}))$. Thus, the distribution $\boldsymbol{y}_i \bmod q\boldsymbol{B}$ is statistically close to uniform over $\mathcal{L}(\boldsymbol{B})/q\mathcal{L}(\boldsymbol{B})$. This shows the statistical closeness of $\boldsymbol{a}_i = (\boldsymbol{B}^{-1}\boldsymbol{y}_i + \boldsymbol{x}_i) \bmod q$ to the uniform over $\mathbb{Z}_q^n$. Since $\boldsymbol{y}_i$ are independent and $m = \mathrm{poly}(n)$, the matrix $\boldsymbol{A}$ is also distributed within negligible distance of the uniform over $\mathbb{Z}_q^{n\times m}$. Hence, O outputs a valid solution $\boldsymbol{e}$ with non-negligible probability.

We can assume that the solution $\boldsymbol{e} \neq \boldsymbol{0}$, $\boldsymbol{e}$ has non-zero coordinate $e_k \in \{-\beta, \ldots, -1, 1, \ldots, \beta\}$ for some $k \in [m]$. This indicates the probability that $j = k$ and $\alpha = e_k$ is negligibly close to $1/(2\beta m) = 1/\mathrm{poly}(n)$ since the reduction algorithm chooses j and α uniformly at random. □

Claim 2.4.7. *If $\boldsymbol{e}$ is a valid solution and $e_j = \alpha$, the output $\boldsymbol{v}$ is a lattice vector of $\mathcal{L}(\boldsymbol{B})$.*

Proof. Notice that $\boldsymbol{v} \in \mathcal{L}(\boldsymbol{B})$ if and only if $\boldsymbol{B}^{-1}\boldsymbol{v} \in \mathbb{Z}^m$. Thus, it suffices to show that $\frac{1}{q}\boldsymbol{B}^{-1}\boldsymbol{Y}\boldsymbol{e} \in \mathbb{Z}^m$, that is, $\boldsymbol{B}^{-1}\boldsymbol{Y}\boldsymbol{e} \in q\mathbb{Z}^m$. By the definition, $\boldsymbol{B}^{-1}\boldsymbol{Y} \equiv \boldsymbol{A} - \boldsymbol{X} \pmod{q}$. Hence, we only need to show that $\boldsymbol{A}\boldsymbol{e} \equiv \boldsymbol{X}\boldsymbol{e} \pmod{q}$. If $e_j = \alpha$, $\boldsymbol{X}\boldsymbol{e} = \alpha \cdot \boldsymbol{x}_j = \boldsymbol{u} \bmod q$. In addition, $\boldsymbol{A}\boldsymbol{e} \bmod q = \boldsymbol{u}$ if $\boldsymbol{e}$ is valid. This completes the proof. □

Claim 2.4.8. *If $\boldsymbol{e}$ is valid and $e_j = \alpha$, then $\|\boldsymbol{v} - \boldsymbol{t}\| \leq \frac{1}{g(n)}\|\boldsymbol{S}\|$ with overwhelming probability.*

Proof. From the hypothesis, we have that $\boldsymbol{t} = \frac{1}{q}\boldsymbol{C}\boldsymbol{e}$, where $\boldsymbol{C} = [\boldsymbol{c}_1, \ldots, \boldsymbol{c}_m]$. For each $\boldsymbol{y}_i$, let $\boldsymbol{w}_i = \boldsymbol{y}_i \bmod q\boldsymbol{B}$ and define $\boldsymbol{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_m]$. Notice that $\boldsymbol{y}_i$ is distributed as $\boldsymbol{w}_i + D_{q\mathcal{L}(\boldsymbol{B}),s,\boldsymbol{c}_i - \boldsymbol{w}_i}$. Note also that the input $(q, \boldsymbol{A}, \boldsymbol{u}, \beta)$ is dependent only $\boldsymbol{W}$ and $\boldsymbol{X}$. Hence, the vector $\boldsymbol{v} - \boldsymbol{t} = \frac{1}{q}(\boldsymbol{Y} - \boldsymbol{C})\boldsymbol{e}$ is distributed as

$$\frac{1}{q}(\boldsymbol{W} - \boldsymbol{C})\boldsymbol{e} + \frac{1}{q}\sum_{i \in [m]} e_i \cdot D_{q\mathcal{L}(\boldsymbol{B}),s,\boldsymbol{c}_i - \boldsymbol{w}_i} = \frac{1}{q}\sum_{i \in [m]} e_i \left(D_{q\mathcal{L}(\boldsymbol{B}),s,\boldsymbol{c}_i - \boldsymbol{w}_i} + \boldsymbol{w}_i - \boldsymbol{c}_i \right).$$

Let $\boldsymbol{z}_i$ be a sample from $D_{q\mathcal{L}(\boldsymbol{B}),s,\boldsymbol{c}_i - \boldsymbol{w}_i}$. So, the vector $\boldsymbol{v} - \boldsymbol{t} = \frac{1}{q}\sum_{i \in [m]} e_i \boldsymbol{z}_i$. Since $s \geq q \cdot \eta_\epsilon(\mathcal{L}(\boldsymbol{B}))$, we can use the lemma 2.1.11 and obtain that the probability

$$\Pr_{\boldsymbol{z}_i \leftarrow D_{q\mathcal{L}(\boldsymbol{B}),s,\boldsymbol{c}_i - \boldsymbol{w}_i}} [\|\boldsymbol{z}_i - (\boldsymbol{c}_i - \boldsymbol{w}_i)\| > s\sqrt{n}]$$

is negligible. Thus, we have each length of $\boldsymbol{z}_i = \boldsymbol{y}_i - \boldsymbol{c}_i$ is at most $s\sqrt{n}$ and by the norm bound, the sum is with overwhelming probability

$$\|\boldsymbol{v} - \boldsymbol{t}\| \leq \frac{1}{q}\sum_{i \in [m]} e_i \|\boldsymbol{z}_i\| \leq \frac{1}{q}\|\boldsymbol{e}\| s\sqrt{n} \leq \frac{\beta\sqrt{n}\|\boldsymbol{S}\|}{\gamma} \leq \frac{\|S\|}{g(n)}.$$

Hence, the norm is upper bounded by $\|\boldsymbol{S}\| / g$ with overwhelming probability. □

The reduction of Micciancio and Regev

In [MR07], Micciancio and Regev gave another reduction from GapSVP to SIS through GapCVP and a variant SIS′ of SIS. Notice that q is slightly larger than the the reduction from SIVP to SIS.

Theorem 2.4.9 (Lemma 5.5 and Theorem 5.23, [MR07]). *For any polynomially bounded functions $\beta = \beta(n)$, $m = m(n)$, odd integer $q = q(n)$, with $q \geq 4\sqrt{m}n^{3/2}\beta$ and $\gamma = \gamma(n) = 14\pi\sqrt{n}\beta$, there exists a probabilistic polynomial-time reduction from solving* GapSVP_γ *in the worst case to solving* $\mathrm{SIS}_{q,m,\beta}$ *on average with non-negligible probability. In particular, for any $m(n) = \Theta(n \log n)$, there exist odd integer $q(n) = O(n^{2.5} \log n)$ and $\gamma(n) = O(n\sqrt{\log n})$ such that solving* $\mathrm{SIS}_{q,m,\sqrt{m}}$ *on average is at least as hard as solving* GapSVP_γ *in the worst case.*

Peikert examined the reduction to $\mathrm{GapSVP}^{p}_{\tilde{O}(n)}$ and it succeeded. For details, see [Pei08]. We note that we have not examined Gentry et al.'s technique can be applied to this reduction and this is an open issue.

2.4.3 From the Learning With Errors

The learning with errors (LWE) problem is a generalization of the learning parity noise (LPN) problem, proposed by Regev [Reg09].

To start the review, we recall the definitions of the distributions appearing the definition of the LWE problem. Later, we define several versions of the LWE problem.

Gaussians: The Gaussian distribution with mean 0 and variance σ^2, denoted by $N(0, \sigma^2)$, is defined by the density function $\frac{1}{\sigma\sqrt{2\pi}} \cdot \exp(-x^2/2\sigma^2)$ over $\mathbb{R}$. By the tail inequality, we have $\Pr[|x| \geq t\sigma] \leq \frac{1}{t} \cdot \exp(-t^2/2)$, where $x \leftarrow N(0, \sigma^2)$.

Folded Gaussians: For $\alpha \in (0, 1)$, Ψ_α denotes the folded Gaussian distribution over $\mathbb{T} = \mathbb{R}/\mathbb{Z}$ [Reg09], obtained by (1) take a sample x from $N(0, \alpha^2/2\pi)$ and (2) output x mod 1. We have $\Pr_{x \leftarrow \Psi_\alpha}[|x| \geq t] \leq \frac{\alpha}{\sqrt{2\pi}t} \cdot \exp(-\pi t^2/\alpha^2)$ by simple calculations. Often, we set t a constant and $\alpha = 1/\omega(\sqrt{\log n})$ to ensure that the right hand side is negligible in n.

Discretized distributions: For any probability distribution ϕ over $\mathbb{T}$ and a positive integer $q \in \mathbb{N}$, $\bar{\phi}$ denotes the discretization of ϕ over $\mathbb{Z}_q$; the distribution is defined by the following procedure, (1) take a sample $x \leftarrow \phi$ and (2) output $\lfloor qx \rceil$ mod q.

The LWE oracle for χ: For $\boldsymbol{s} \in \mathbb{Z}_q^n$ and a distribution χ over $\mathbb{T}$, let $A_{\boldsymbol{s},\chi}$ be a distribution over $\mathbb{Z}_q^n \times \mathbb{T}$ defined as follows: (1) take samples $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ and $x \leftarrow \chi$ and (2) output $(\boldsymbol{a}, \boldsymbol{a}^T\boldsymbol{s}/q + x)$.

The LWE oracle for $\bar{\chi}$: For $\boldsymbol{s} \in \mathbb{Z}_q^n$ and a distribution $\bar{\chi}$ over $\mathbb{Z}_q$, let $A_{\boldsymbol{s},\bar{\chi}}$ be a distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ defined as follows: (1) take samples $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ and $x \leftarrow \bar{\chi}$ and (2) output $(\boldsymbol{a}, \boldsymbol{a}^T\boldsymbol{s} + x)$.

For simplifying expressions, we define $A_{\boldsymbol{S},\bar{\chi}}$ for a matrix $\boldsymbol{S} \in \mathbb{Z}_q^{n\times l}$ as follows: (1) take samples $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{x} \leftarrow \bar{\chi}^l$ and (2) output $(\boldsymbol{a}, \boldsymbol{a}^T\boldsymbol{S} + \boldsymbol{x})$.

We define the learning with errors (LWE) problem as follows:

Definition 2.4.10 ((Search) Learning With Errors)**.** The (search) LWE problem with respect to q and χ, denoted by sLWE(q,χ), is finding $\boldsymbol{s} \in \mathbb{Z}_q^n$ given oracle access to $A_{\boldsymbol{s},\chi}$. The (search) LWE problem with respect to q and $\bar{\chi}$, denoted by sLWE$(q,\bar{\chi})$, is finding $\boldsymbol{s} \in \mathbb{Z}_q^n$ given oracle access to $A_{\boldsymbol{s},\bar{\chi}}$.

Definition 2.4.11 ((Decisional) Learning With Errors)**.** For an integer $q = q(n)$ and a distribution $\bar{\chi}$ over $\mathbb{Z}_q$, the (decision) learning with errors problem dLWE$(q,\bar{\chi})$ is distinguishing the oracle $A_{\boldsymbol{s},\chi}$ from the oracle $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ for a uniformly random $\boldsymbol{s} \in \mathbb{Z}_q^n$.

In addition, for an integer $q = q(n)$ and a distribution χ over $\mathbb{T}$, the (decision) learning with errors problem dLWE$(q,\bar{\chi})$ is distinguishing the oracle $A_{\boldsymbol{s},\chi}$ from the oracle $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ for a uniformly random $\boldsymbol{s} \in \mathbb{Z}_q^n$.

These problems are closely related to the decoding problem with a random linear code [Reg09]. Consider m samples $(\boldsymbol{a}_i, p_i = \langle \boldsymbol{a}_i, \boldsymbol{s}\rangle + x_i)$ from $A_{\boldsymbol{s},\bar{\chi}}$. These can be considered as $(\boldsymbol{A}, \boldsymbol{p})$, where $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_m]$ and $\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x}$. The sLWE$(q,\bar{\chi})$ problem can be stated as coding problem as follows: Given a random generator matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x}$, where $\boldsymbol{x} \leftarrow \bar{\chi}^l$, the problem is decoding $\boldsymbol{s}$. In addition, the dLWE$(q,\bar{\chi})$ also can be stated as coding problem; given a random generator matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{p}$, the problem is deciding $\boldsymbol{p}$ is random or not.

Note that an adversary $\mathcal{A}$ distinguishing $\boldsymbol{A}_{s,\chi}$ and $U(\mathbb{Z}_q^n \times \mathbb{Z}_q)$ with advantage ϵ implies an adversary distinguishing $\boldsymbol{A}_{\boldsymbol{S},\chi}$ and $U(\mathbb{Z}_q^n \times \mathbb{Z}_q^l)$ for $\boldsymbol{S} \leftarrow \mathbb{Z}_q^l$ with advantage ϵ/l. The proof is simply obtained by the hybrid lemma [PVW08].

Main Reduction

These LWE problems already has the average-case/worst-case reductions. We start with the reduction from the worst case of dLWE$(q, \bar{\chi})$ to the average case of dLWE$(q, \bar{\chi})$.

Lemma 2.4.12 (Average case to Worst case [Reg09])**.** *Suppose that there is an algorithm $\mathcal{A}$ that distinguishes $A_{\boldsymbol{s},\bar{\chi}}$ from U in time T and with noticeable advantage ϵ, where the probability is taken over the coin of the algorithm, the samples from the oracle, and $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$. Then, there is an algorithm $\mathcal{B}$ that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, distinguishes $A_{\boldsymbol{s},\bar{\chi}}$ from U in time* $\mathrm{poly}(T, n, \log q, 1/\epsilon)$ *with overwhelming probability.*

Proof Sketch. For $\boldsymbol{s}' \in \mathbb{Z}_q^n$, we define the mapping $T_{\boldsymbol{s}'} : \mathbb{Z}_q^n \times \mathbb{Z}_q \to \mathbb{Z}_q^n \times \mathbb{Z}_q$ by $T_{\boldsymbol{s}'}(\boldsymbol{a}, p) = (\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}' \rangle + p)$. Obviously, $T_{\boldsymbol{s}'}$ maps $A_{\boldsymbol{s},\bar{\chi}}$ and U to $A_{\boldsymbol{s}+\boldsymbol{s}',\bar{\chi}}$ and U, respectively. The lemma immediately follows from this random self reducibility. □

Lemma 2.4.13 (Decision to Search [Reg09])**.** *Suppose that* $q = \mathrm{poly}(n)$*. Then, if there is an algorithm that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, distinguishes $A_{\boldsymbol{s},\bar{\chi}}$ from U in time T and with overwhelming probability , then there exists an algorithm that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, finds $\boldsymbol{s}$ from $A_{\boldsymbol{s},\bar{\chi}}$ in time* $\mathrm{poly}(T, n, q)$ *and with overwhelming probability.*

Proof Sketch. We construct an indicator $\mathcal{I}$ which outputs the j-th coordinate s_j of $\boldsymbol{s}$ from $A_{\boldsymbol{s},\bar{\chi}}$ with oracle access to the distinguisher $\mathcal{D}$. For any $k \in \mathbb{Z}_q$, we define a random mapping T_k defined by

$$T_k(\boldsymbol{a}, p) = (\boldsymbol{a} + l \cdot \boldsymbol{i}_j, p + lk)$$

where $l \leftarrow \mathbb{Z}_q$. If $k = s_j$, T_k maps $A_{\boldsymbol{s},\bar{\chi}}$ to itself. Otherwise, T_k maps $A_{\boldsymbol{s},\bar{\chi}}$ to U since q is a prime. Since q is polynomially-bounded by n, the finding s_j is straightforward; examine all $k \in \mathbb{Z}_q$. □

The following reduction is obvious if we take a precision with $\omega(\log n)$.

Lemma 2.4.14 (Discrete to Continuous [Reg09])**.** *For any q and χ, if there exists an algorithm that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, finds $\boldsymbol{s}$ from $A_{\boldsymbol{s},\bar{\chi}}$ in time T and with overwhelming probability, then there exists an algorithm that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, finds $\boldsymbol{s}$ from $A_{\boldsymbol{s},\chi}$ in time $O(T, \log q, n)$ and with overwhelming probability.*

Even if q is not polynomially-bounded, there is a decision-to-search reduction when q is a product of distinct polynomially-bounded primes.

Lemma 2.4.15 (Decision to Search [Pei08])**.** *Suppose that $q = \prod_{i \in [t]} q_i$ is a product of distinct primes in n. Suppose also $\alpha = \alpha(n) \in (0, 1)$ be a real such that*

$q_i \geq \omega(\sqrt{\log n})/\alpha$ for any $i \in [t]$. Then, if there is an algorithm that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, distinguishes $A_{\boldsymbol{s},\Psi_\alpha}$ from U in time T and with overwhelming probability , then there exists an algorithm that, for any $\boldsymbol{s} \in \mathbb{Z}_q^n$, finds $\boldsymbol{s}$ from $A_{\boldsymbol{s},\Psi_\alpha}$ in time $\mathrm{poly}(T, n, t, \max_i q_i)$ *and with overwhelming probability.*

The proof appeared in [Pei08] and he wrote that the idea is due to Regev.

Proof. Obviously, we can factor q into $\prod_i q_i$ in time $\mathrm{poly}(t, \max_i q_i)$. By shifting $\boldsymbol{s}$ as in the average-to-worst reduction, we only need a power to decide $s_j \equiv 0 \bmod q_i$ for any $i \in [t]$ and $j \in [n]$.

Define the random mapping

$$T : (\boldsymbol{a}, p) \mapsto (\boldsymbol{a} - r \cdot (q/q_i) \cdot \boldsymbol{i}_j, p)$$

where $r \leftarrow \mathbb{Z}_{q_i}$. If $s_j \equiv 0 \mod q_i$, T maps $A_{\boldsymbol{s},\bar{\Psi}_\alpha}$ to itself. Suppose that $s_j \not\equiv 0 \bmod q_i$. Then, obviously $\boldsymbol{a}' = \boldsymbol{a} - r \cdot (q/q_i) \cdot \boldsymbol{i}_j$ is uniformly random. We have that

$$p' = p = \langle \boldsymbol{a}, \boldsymbol{s} \rangle / q + x = \langle \boldsymbol{a}', \boldsymbol{s} \rangle / q + (r s_j / q_i + x) \in \mathbb{T},$$

where $x \leftarrow \Psi_\alpha$. Since q_i is a prime, $r s_j$ is uniformly distributed over $\mathbb{Z}_{q_i}$. Since $\alpha \geq \omega(\sqrt{\log n})/q_i \geq \eta_\epsilon(\frac{1}{q_i}\mathbb{Z})$ for some negligible ϵ, the distribution of $r s_j / q_i + e \bmod 1$ is within $\epsilon/2$ statistical distance from uniform over $\mathbb{T}$ by Lemma 2.1.4. This completes the proof. □

Finally, we recall the theorems by Regev [Reg09] and Peikert [Pei08].

Theorem 2.4.16 (Regev [Reg09] and Peikert [Pei08])**.** *Let n, $m = m(n)$, $q = q(n)$ be integers and $\alpha = \alpha(n) \in (0, 1)$ be such that $\alpha q > 2\sqrt{n}$, $m = \mathrm{poly}(n)$, and $q = 2^{O(n)}$. If there exists an efficient algorithm that solves* $\mathrm{sLWE}(q, m, \Psi_\alpha)$*, then there exists an efficient* quantum *algorithm that solves* GapSVP_γ *and* SIVP_γ *with $\gamma = \tilde{O}(n/\alpha)$ in the worst case.*

Alternatively, let n and $m = m(n) = \mathrm{poly}(n)$ be integers. Let $\alpha = \alpha(n)$ be a real number and $\gamma = \gamma(n) \geq n/(\alpha\sqrt{\log n})$. Let $\zeta = \zeta(n) \geq \gamma$ and $q = q(n) \geq (\zeta/\sqrt{n}) \cdot \omega(\sqrt{\log n})$. Then, there is a probabilistic polynomial-time reduction from solving $\mathrm{GapSVP}_{\zeta,\gamma}$ *in the worst case to solving* $\mathrm{sLWE}(q, m, \Psi_\alpha)$*.*

Regev's Reduction

We first review Regev's reduction from DGS to $\mathrm{sLWE}(q, \Psi_\alpha)$. The reduction is divided into twofolds. The classical reduction is from $\mathrm{sLWE}(q, \Psi_\alpha)$ and a sampling problem with respect to $D_{\Lambda,s}$ to CVP on Λ^*. The quantum reduction is from CVP to the sampling problem with respect to $D_{\Lambda,s}$. We only give intuitions on the quantum part of the reduction.

Classical part: More precisely, the former reduction is described as follows:

Theorem 2.4.17. *Let $\epsilon = \epsilon(n)$ be a negligible function, $q = q(n) \geq 2$ be an integer, $\alpha = \alpha(n) \in (0, 1)$. If there exist algorithms that solve* $\mathrm{sLWE}_{q,\Psi_\alpha}$ *and* DGS_ϕ *for $\phi = \sqrt{2}q \cdot \eta_\epsilon$, then there exists an algorithm that solves* $\mathrm{BDD}_{\alpha q/(\sqrt{2}r)}$.

Precisely, there is a classical probabilistic polynomial-time algorithm $\mathcal{R}(\boldsymbol{B}, r, \boldsymbol{x})$ that, given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ, a number $r \geq \sqrt{2}q \cdot \eta_\epsilon(\Lambda^*)$, and a target point $\boldsymbol{x}$ within distance $\alpha q/(\sqrt{2}r)$ of Λ, and given access to

1. *an oracle $\mathcal{W}$ that solves* $\mathrm{sLWE}_{q,\Psi_\alpha}$ *using* $\mathrm{poly}(n)$ *samples, and*
2. *an oracle $\mathcal{S}$ that generates samples from $D_{\Lambda^*,r}$,*

finds $\boldsymbol{v} \in \Lambda$ closest to $\boldsymbol{x}$ with overwhelming probability.

We start by defining a technical problem $\mathrm{BDD}^{(q)}$ which has a connection to sLWE.

Definition 2.4.18. For $q \geq 2$, the problem $\mathrm{BDD}^{(q)}$ is, given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ and a number $d < \lambda_1(\Lambda)/2$, finding $\boldsymbol{w} \bmod q \in \mathbb{Z}_q^n$ such that $\boldsymbol{B}\boldsymbol{w}$ is the unique closest vector to $\boldsymbol{x}$.

Regev showed the following reduction.

Lemma 2.4.19 (Lemma 3.5, [Reg09])**.** *There is a lattice-preserving reduction from* BDD_d *to* $\mathrm{BDD}_d^{(q)}$ *if $d < \lambda_1(\Lambda)/2$.*

Since $\alpha q/(\sqrt{2}r) \leq \lambda_1(\Lambda)/2$, it is suffices to construct $\mathcal{R}'$ that solves $\mathrm{BDD}^{(q)}_{\alpha q/(\sqrt{2}r)}$ with help of the oracles $\mathcal{S}$ and $\mathcal{W}$.

Let $\boldsymbol{v}$ be a solution of $(\boldsymbol{B}, r, \boldsymbol{x})$. We let $\boldsymbol{s}$ denote $\boldsymbol{B}^{-1}\boldsymbol{v} \bmod q$. In order to generate a sample from $A_{\boldsymbol{s},\Psi_\alpha}$, take a sample $\boldsymbol{y}$ from $D_{\Lambda^*,r}$, lets $\boldsymbol{a} = (\boldsymbol{B}^*)^T\boldsymbol{y} \bmod q$, and outputs

$$(\boldsymbol{a}, p = \langle \boldsymbol{y}, \boldsymbol{x}\rangle/q + x \bmod 1),$$

where $\boldsymbol{x} \leftarrow N(0, \alpha^2/4\pi)$. By the construction, we have that

$$\begin{aligned}\langle \boldsymbol{y}, \boldsymbol{x}\rangle/q + x &= \langle \boldsymbol{y}, \boldsymbol{v}\rangle/q + \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{v}\rangle/q + x \\ &= \langle \boldsymbol{B}^T\boldsymbol{y}, \boldsymbol{B}^{-1}\boldsymbol{v}\rangle + \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{v}\rangle/q + x \\ &\equiv \langle \boldsymbol{a}, \boldsymbol{s}\rangle + \langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{v}\rangle/q + x \bmod 1.\end{aligned}$$

As we already seen in the reduction to SIS, $\boldsymbol{a}$ is almost uniformly distributed over $\mathbb{Z}_q^n$, since $r \geq \sqrt{2}q \cdot \eta_\epsilon(\Lambda^*)$.

To ensure that $\langle \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{v}\rangle/q + x \bmod 1$ distributes as a continuous Gaussian, we use the following claim that says that the sum of a Gaussian over a lattice and a continuous Gaussian distributes statistically close to another continuous Gaussian.

Claim 2.4.20 (Claim 3.9, [Reg09])**.** *Let Λ be a lattice and $\boldsymbol{u} \in \mathbb{R}^n$. Let r and s be two positive integers, and let t denote $\sqrt{r^2 + s^2}$. Suppose that $rs/t =\geq \eta_\epsilon(\Lambda)$ for some $\epsilon \in (0, 1/2)$. Consider the distribution Y on $\mathbb{R}^n$ defined as follows: (1) take* *a sample $\boldsymbol{y} \leftarrow D_{\Lambda+\boldsymbol{u},r}$, (2) take a noise $\boldsymbol{x} \leftarrow \nu_s$, and (3) output $\boldsymbol{y}+\boldsymbol{x}$. Then, the statistical distance between Y and ν_t is at most 4ϵ.*

Following the claim, we obtain the corollary below.

Corollary 2.4.21 (Corollary 3.10, [Reg09])**.** *Let Λ be a lattice and $\boldsymbol{u}, \boldsymbol{z} \in \mathbb{R}^n$ vectors. Let $r, \alpha > 0$ be two reals. Suppose that*

$$\frac{1}{\sqrt{\frac{1}{r^2} + \left(\frac{\|\boldsymbol{z}\|}{\alpha}\right)^2}} \geq \eta_\epsilon(\Lambda)$$

for some $\epsilon \in (0, 1/2)$. Let B denote the distribution sampled as follows: (1) $\boldsymbol{v} \leftarrow \boldsymbol{u} + D_{\Lambda,r,-\boldsymbol{u}}$, (2) $x \leftarrow N(0, \alpha^2/2\pi)$, and (3) output $\langle \boldsymbol{z}, \boldsymbol{v}\rangle + x$. Then, we have

$$\Delta(B, N(0, (r^2 \|\boldsymbol{z}\|^2 + \alpha^2)/2\pi)) \leq 4\epsilon.$$

In particular,

$$\Delta(B \bmod 1, \Psi_\beta) \leq 4\epsilon,$$

where $\beta = \sqrt{(r\|\boldsymbol{z}\|)^2 + \alpha^2}$.

Conditioned on $\boldsymbol{a}$, the distribution of $\boldsymbol{y}$ is $\boldsymbol{Ba} + D_{q\Lambda,r,-\boldsymbol{Ba}}$. In addition, we have that

$$\begin{aligned}\frac{1}{\sqrt{\frac{1}{r^2} + \left(\frac{\sqrt{2}\|\boldsymbol{x}-\boldsymbol{v}\|}{q\alpha}\right)^2}} &= \frac{rq\alpha}{\sqrt{q^2\alpha^2 + 2r^2\|\boldsymbol{x}-\boldsymbol{v}\|^2}} \\ &\geq \frac{r}{\sqrt{2}} > q \cdot \eta_\epsilon(\Lambda) = \eta_\epsilon(q\Lambda),\end{aligned}$$

where we use $\|\boldsymbol{x}-\boldsymbol{v}\| \leq q\alpha/(\sqrt{2}r)$. Now, by the corollary, $\langle \boldsymbol{y}, \boldsymbol{x}-\boldsymbol{v}\rangle/q + x \bmod 1$ is statistically close to $\Psi_{\alpha'}$ for some $\alpha' = \sqrt{(r\|\boldsymbol{x}-\boldsymbol{v}\|/q)^2 + \alpha^2/2} \leq \alpha$. Since the solver $\mathcal{W}$ also solves $\mathrm{sLWE}_{q,\Psi_{\alpha'}}$ when $\alpha' \leq \alpha$ (see [Reg09, Lemma 3.7]), we can recover $\boldsymbol{s}$ by the oracle $\mathcal{W}$ for $\mathrm{sLWE}(q, \Psi_\alpha)$.

Quantum part: Turning into the latter reduction, Regev constructed the *quantum* sampler $\mathcal{S}$ for $D_{\Lambda,\sqrt{n}/(\sqrt{2}d)}$ from BDD_d.

Theorem 2.4.22 ([Reg09, Lemma 3.14])**.** *There exists an efficient quantum algorithm that, given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ, a number $d < \lambda_1(\Lambda^*)/2$, and an oracle that solves BDD_d on a dual lattice Λ^*, outputs a sample from $D_{\Lambda,\sqrt{n}/(\sqrt{2}d)}$.*

Intuitively, BDD_d can be used to *uncompute*; that is, $|\boldsymbol{x}\rangle|\boldsymbol{x}+\boldsymbol{e}\rangle \mapsto |\boldsymbol{0}\rangle|\boldsymbol{x}+\boldsymbol{e}\rangle$ for any $\boldsymbol{x} \in \Lambda^*$ and $\boldsymbol{e}$ with norm at most d. We note that this reduction is lattice-preserving. This will be exploited in Section 3.4.2.

Combining them: The bootstrapping step is done by the LLL algorithm and the sampler SampleD (see Section 10.4). Given a basis $\boldsymbol{B}$ of Λ, we obtain a new basis $\boldsymbol{T}$ that $\|\boldsymbol{T}\| \leq 2^n \lambda_n(\Lambda)$ applying the LLL algorithm. Hence, if $r \geq \|\boldsymbol{T}\| \cdot \omega(\sqrt{\log n})$, we can use the sampler SampleD to sample from $D_{\Lambda,r}$.

The whole reduction from sLWE to DGS is summarized as follows: Suppose that the algorithm $\mathcal{R}$ needs m samples from $\mathcal{S}$ to invoke $\mathcal{W}$.

1. (*bootstrapping*) Apply the LLL algorithm to an input basis $\boldsymbol{B}$ of a lattice Λ and obtain $\boldsymbol{T}$. Set $r \geq \|\boldsymbol{T}\| \cdot \omega(\sqrt{\log n})$. Construct $\mathcal{S}$ for $D_{\Lambda,r}$ by SampleD with $\boldsymbol{T}$ and r.
2. (*iterative step*)
   (a) Construct the algorithm $\mathcal{R}$ for BDD_d on Λ^* and $d = \alpha q/(\sqrt{2}r)$ by using the oracle $\mathcal{W}$ and the algorithm $\mathcal{S}$ for $D_{\Lambda,r}$.
   (b) Construct the quantum sampler $\mathcal{S}'$ for $D_{\Lambda,\sqrt{n}/(\sqrt{2}d)}$ by using the algorithm $\mathcal{R}$ for BDD_d on Λ^* and $d = \alpha q/\sqrt{2}r$. Note that $\sqrt{n}/(\sqrt{2}d) = \sqrt{n}r/\alpha q \leq r/2$ since $\alpha q > 2\sqrt{n}$.

When $r < \sqrt{2}q \cdot \eta_\epsilon(\Lambda)$, the algorithm will fail.

To connect DGS_ϕ and SIVP_γ is somewhat simpler task than the aboves. See [Reg09] for the proof.

Lemma 2.4.23 ([Reg09, Lemma 3.17]). *For any $\epsilon = \epsilon(n) \leq 1/10$ and any $\phi \geq \sqrt{2}\eta_\epsilon$, there is a lattice-preserving polynomial-time reduction from $\mathrm{GIVP}_{2\sqrt{n}\phi}$ to DGS_ϕ.*

Peikert's Reduction

The classical part of Regev's reduction is from $\mathrm{sLWE}_{q,\Psi_\alpha}$ and the sampler $\mathcal{S}$ for $D_{\Lambda^*,r}$ to BDD_d over Λ, where $d = \alpha q/(\sqrt{2}r) \leq \lambda_1(\Lambda)/2$.

Peikert pointed out the existence of the sampler SampleD for $D_{\Lambda^*,r}$ when r is slightly larger than the norm of a basis $\boldsymbol{B}$. In addition, he recalled the result of Goldreich and Goldwasser [GG00] that GapCVP_γ is in coAM when $\gamma = O(\sqrt{n/\log n})$.

The nutshell of [GG00] is the observation on two balls:

Lemma 2.4.24. *For any constants $c, d > 0$, and any $\boldsymbol{u} \in \mathbb{R}^n$ of length d, and $d' = d \cdot \sqrt{n/(c \log n)}$,*

$$\Delta(U(B(\mathbf{0}, d')), U(B(\boldsymbol{u}, d'))) \leq 1 - 1/\operatorname{poly}(n).$$

This lemma states that if $\boldsymbol{x} \leftarrow B(\mathbf{0}, d')$, it will be contained in $B(\boldsymbol{u}, d')$ with probability at least $1/\operatorname{poly}(n)$. Hence, one cannot distinguish $\boldsymbol{x}$ is chosen from $B(\mathbf{0}, d')$ or $B(\boldsymbol{u}, d')$ conditioned on that $\boldsymbol{x}$ is in both balls. In addition, notice that, if $d' \geq \lambda_1(\Lambda)/2$, the two balls do not overlap.

We give the details of the Peikert reduction [Pei09c]. The input is $(\boldsymbol{B}, d)$ where $\min_i \|\tilde{\boldsymbol{b}}_i\| \geq 1$, $\lambda_1(\Lambda) \leq \zeta$, and $1 \geq d \geq \zeta/\gamma$.

1. Set $r = q \cdot \sqrt{2n}/(\gamma d)$.
2. Implement the oracle $\mathcal{S}$ for $D_{\Lambda^*,r}$ by the sampler SampleD with the dual basis $\boldsymbol{B}^*$.
3. Repeat the following procedure $N = \mathrm{poly}(n)$ times.
   (a) Choose a point $\boldsymbol{w} \leftarrow B(\boldsymbol{0}, d')$, where $d' = d \cdot \sqrt{n/(4\log n)}$, and let $\boldsymbol{x} = \boldsymbol{w} \bmod B$.
   (b) Invoke the reduction $\mathcal{R}$ on $(\boldsymbol{B}, r, \boldsymbol{x})$ and obtain $\boldsymbol{v}$.
4. If $\boldsymbol{v} \neq \boldsymbol{x} - \boldsymbol{w}$ in any of the N iterations, then accept. Otherwise, reject.

Notice that $\max_i \|\tilde{\boldsymbol{b}}_i^*\| = 1/\min_i \|\tilde{\boldsymbol{b}}_i\| \leq 1$. Notice also that

$$r = \frac{q\sqrt{2n}}{\gamma d} \geq \frac{q\sqrt{2n}}{\zeta} \geq \omega(\sqrt{\log n}),$$

since $q \geq (\zeta/\sqrt{2n}) \cdot \omega(\sqrt{\log n})$.

If $(\boldsymbol{B}, d)$ is a NO instance, then, $\lambda_1(\Lambda) > \gamma \cdot d$. Lemma 2.1.6 tells us that

$$\eta_\epsilon(\Lambda^*) \leq \sqrt{n}/\lambda_1(\Lambda) < \sqrt{n}/(\gamma d)$$

for $\epsilon = 2^{-n}$. Thus, we have that $\sqrt{2}q \cdot \eta_\epsilon(\Lambda^*) < \sqrt{2n}q/(\gamma d) \leq r$. Additionally, we have that

$$\mathrm{dist}(\boldsymbol{x}, \Lambda) \leq d' = d\sqrt{\frac{n}{4\log n}} \leq d \cdot \frac{\alpha\gamma}{\sqrt{4n}} = \frac{q\sqrt{2n}}{r \cdot \gamma} \cdot \frac{\alpha\gamma}{\sqrt{4n}} = \frac{\alpha q}{\sqrt{2}r},$$

where we use the hypothesis $\gamma \geq n/(\alpha\sqrt{\log n})$. From these two facts, the reduction $\mathcal{R}$ correctly works. Since $\lambda_1(\Lambda) > \gamma d > 2d'$, the reduction must return the unique solution $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{w}$ in any iterations.

Next, consider the case when $(\boldsymbol{B}, d)$ is a YES instance ($\lambda_1(\Lambda) \leq d$). Notice that in the case, we cannot ensure that the reduction $\mathcal{R}$ correctly works. However, we can show the reduction $\mathcal{R}$ fails to finds the solution $\boldsymbol{v} = \boldsymbol{x} - \boldsymbol{w}$ by the argument on the statistical distance. Let $\boldsymbol{z} \in \Lambda$ be the shortest vector, that is, $\|\boldsymbol{z}\| = \lambda_1(\Lambda)$. Consider an alternate game in which of $\boldsymbol{w} \leftarrow B(\boldsymbol{0}, d')$ is replaced by $\boldsymbol{w}' \leftarrow B(\boldsymbol{z}, d')$. We then replace $\boldsymbol{x} = \boldsymbol{w} \bmod \boldsymbol{B}$ with $\boldsymbol{x}' = \boldsymbol{w}' \bmod \boldsymbol{B}$. In this alternate game, $\mathcal{R}$ is invoked on $\boldsymbol{x}'$. Then, we have that

$$|\Pr[\mathcal{R}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{w}] - \Pr[\mathcal{R}(\boldsymbol{x}') = \boldsymbol{x}' - \boldsymbol{w}']| \leq 1 - \frac{1}{\mathrm{poly}(n)}.$$

Hence,

$$\Pr[\mathcal{R}(\boldsymbol{x}) = \boldsymbol{x}-\boldsymbol{w}] \leq 1 - \frac{1}{\mathrm{poly}(n)} + \Pr[\mathcal{R}(\boldsymbol{x}') = \boldsymbol{x}'-\boldsymbol{w}'] \leq 2 - \frac{1}{\mathrm{poly}(n)} - \Pr[\mathcal{R}(\boldsymbol{x}') = \boldsymbol{x}'-\boldsymbol{w}'].$$

Notice that $B(\boldsymbol{z}, d') \equiv B(\boldsymbol{0}, d') \pmod{\boldsymbol{B}}$ since $\boldsymbol{z} \in \Lambda$. Thus, $\boldsymbol{x}'$ is distributed identically to $\boldsymbol{x}$ and we can replace $\boldsymbol{x}$ with $\boldsymbol{x}'$ in the probabilities. Therefore,

$$\Pr[\mathcal{R}(\boldsymbol{x}) = \boldsymbol{x} - \boldsymbol{w}] \leq 1 - \frac{1}{2 \cdot \mathrm{poly}(n)}.$$

Taking $N = \text{poly}(n)$ sufficiently large, we have $\boldsymbol{v} \neq \boldsymbol{x} - \boldsymbol{w}$ in at least one iteration.

3 Cyclic and Ideal Lattices

Several families of lattice-based hash functions [Mic07, PR06, LM06] are known to have small description sizes. Originally, Micciancio [Mic07] gave a compact version of the lattice-based hash functions and proved the one-wayness of the version. After that, Peikert and Rosen [PR06] and Lyubashevsky and Micciancio [LM06] proposed the modified versions of the version of Micciancio and showed their collision-resistance property, independently. The underlying problems are lattice problems whose instances are lattices has certain algebraic structure and compact description, *cyclic or ideal lattices*. We employ the notions, the notations, the definitions, and the results in Lyubashevsky and Micciancio [LM06], since its generality of the descriptions.

Organization: Section 3.1 prepares basic notions, notation, and facts of polynomials. Section 3.2 reviews and defines several lattices. Section 3.3 lists up the problems for ideal lattices. Section 3.4 reviews the average-case/worst-case reductions from ideal-lattice versions of SIS and LWE to ideal lattice problems.

3.1 Preliminaries

Let $\mathbf{f}(x) = f_0 + f_1 x + \cdots + f_{n-1} x^{n-1} + x^n \in \mathbb{Z}[x]$ be a monic polynomial of degree n. Consider the quotient ring $R_{\mathbf{f}} = \mathbb{Z}[x]/\langle \mathbf{f} \rangle$. We use the standard set of representatives $\{(\mathbf{g} \bmod \mathbf{f}) \mid \mathbf{g} \in \mathbb{Z}[x]\}$ for $R_{\mathbf{f}}$. Hence, we identify an integral polynomial $\mathbf{a}$ of degree at most $n-1$ with the corresponding representative $(\mathbf{a} \bmod \mathbf{f})$. In addition, we identify a polynomial $\mathbf{a}(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \in R_{\mathbf{f}}$ with an n-dimensional integer vector $\boldsymbol{a} = (a_0, \ldots, a_{n-1}) \in \mathbb{Z}^n$ in this thesis. More precisely, consider the following mapping $\sigma : R_{\mathbf{f}} \to \mathbb{Z}^n$:

$$\sigma : a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} \mapsto (a_0, a_1, \ldots, a_{n-1}).$$

We call this embedding as the normal embedding of $R_{\mathbf{f}}$ into $\mathbb{Z}^n$. We omit this σ unless we need it explicitly.

Let us define some useful functions and examine these properties. A function $\mathrm{rot}_{\mathbf{f}} : R_{\mathbf{f}} \to R_{\mathbf{f}}$ is defined by $\mathrm{rot}_{\mathbf{f}}(\mathbf{a}) = \mathbf{a} \otimes x$ (naturally extended to $\mathrm{rot}_{\mathbf{f}} : \mathbb{Q}[x]/\langle \mathbf{f} \rangle \to \mathbb{Q}[x]/\langle \mathbf{f} \rangle$ and $\mathrm{rot}_{\mathbf{f}} : \mathbb{Q}^n \to \mathbb{Q}^n$ by σ). This function is linear because the map is represented by the matrix

$$\boldsymbol{R}_{\mathbf{f}} = \begin{pmatrix} 0 & 0 & \ldots & 0 & -f_1 \\ 1 & 0 & \ldots & 0 & -f_2 \\ 0 & 1 & & 0 & \vdots \\ 0 & 0 & \ddots & 0 & -f_{n-2} \\ 0 & 0 & \ldots & 1 & -f_{n-1} \end{pmatrix}.$$

By the identification σ, we have

$$\mathrm{rot}_{\mathbf{f}}(\mathbf{a}) = \boldsymbol{R}_{\mathbf{f}} \cdot \boldsymbol{a}.$$

A function $\mathrm{Rot}_{\mathbf{f}} : R_{\mathbf{f}} \to \mathbb{Z}^{n \times n}$ is defined by $\mathrm{Rot}_{\mathbf{f}}(\mathbf{a}) = [\sigma(\mathbf{a}), \sigma(\mathrm{rot}_{\mathbf{f}}(\mathbf{a})), \ldots, \sigma(\mathrm{rot}_{\mathbf{f}}^{n-1}(\mathbf{a}))]$. For example, if $\mathbf{f} = x^n - 1$ or $\mathbf{f} = x^n + 1$, $\mathrm{Rot}_{\mathbf{f}}(\mathbf{a})$ is a circulant or nega-circulant matrix, respectively. We next define a ring of matrices corresponding to polynomials in $R_{\mathbf{f}}$. Let $\mathcal{M} = \{\mathrm{Rot}_{\mathbf{f}}(\mathbf{a}) \in \mathbb{Z}^{n \times n} \mid \mathbf{a} \in R_{\mathbf{f}}\}$. Then, $\mathrm{Rot}_{\mathbf{f}} : R_{\mathbf{f}} \to \mathcal{M}$ is a ring isomorphism, since $\mathrm{Rot}_{\mathbf{f}}$ is homomorphic and one to one. Additionally, notice that

$$\mathbf{a} \otimes \mathbf{b} = \mathrm{Rot}_{\mathbf{f}}(\mathbf{a}) \cdot \boldsymbol{b}.$$

In addition, we note that the above arguments are also applied if $\mathbb{Z}$ is replaced with $\mathbb{Z}_q$ for any integer $q \geq 2$.

We define a norm with respect to $\mathbf{f}$ as follows: For $\mathbf{g} \in \mathbb{Z}[x]$, $||\mathbf{g} + \langle \mathbf{f} \rangle||_{\mathbf{f}} = ||(\mathbf{g} \bmod \mathbf{f})||_\infty$. We write $||\mathbf{g}||_{\mathbf{f}}$ instead of $||\mathbf{g} + \langle \mathbf{f} \rangle||_{\mathbf{f}}$.

The property of $\mathbf{f}$ is defined as that the ring norm $||\mathbf{g}||_{\mathbf{f}}$ is not much bigger than $||\mathbf{g}||_\infty$ for any polynomial $\mathbf{g}$. Formally, they captured this property as the *expansion factor* of $\mathbf{f}$:

$$\mathrm{EF}_\infty(\mathbf{f}, k) = \max_{\mathbf{g} \in \mathbb{Z}[x], \deg(\mathbf{g}) \leq k(\deg(\mathbf{f})-1)} ||\mathbf{g}||_{\mathbf{f}} \,/\, ||\mathbf{g}||_\infty .$$

For example, a simple calculation shows that $\mathrm{EF}(x^n \pm 1, k) \leq k$ and $\mathrm{EF}(x^{n-1} + x^{n-2} + \cdots + 1, k) \leq 2k$. We say a polynomial $\mathbf{f}$ is suitable if $\mathbf{f}$ is a monic and irreducible in $\mathbb{Z}[x]$ and there is a constant c such that $\mathrm{EF}(\mathbf{f}, k) \leq ck$ for any natural number k. See [LM06, Section 3.1] for more details. They employed a family of polynomials such as $x^n + 1$ and $x^{n-1} + x^{n-2} + \cdots + 1$ for n such that the polynomials are irreducible in $\mathbb{Z}[x]$.

Note that the relation of $\mathbf{f} = x^n + 1$ and q, which will be exploited later.

Lemma 3.1.1. *Let* $\mathbf{f} = x^n + 1$ *and* $n = 2^k$*, where* $k \geq 2$*. If* q *is a prime with* $2n|(q-1)$*, then* $\mathbf{f} = \prod_{i\in[n]}(x - w^{2i+1})$ *over* $\mathbb{Z}_q$*, where* $w \in \mathbb{Z}_q^*$ *is a generator of a subgroup* $\{w^0, w^1, \dots\} \subseteq \mathbb{Z}_q^*$*of cardinality* 2^k*.*

Lemma 3.1.2 ([BGM93])**.** *Let* $\mathbf{f} = x^n + 1$ *and* $n = 2^k$*, where* $k \geq 2$*. If* q *is a prime with* $q \equiv 3 \pmod 8$*,* $\mathbf{f} = \mathbf{f}_1 \cdot \mathbf{f}_2$ *over* $\mathbb{Z}_q$*, where each* $\mathbf{f}_i$ *is irreducible in* $\mathbb{Z}_q[x]$ *and can be written* $\mathbf{f}_i = x^{n/2} + t_i x^{n/4} - 1$ *with* $t_i \in \mathbb{Z}_q$*.*

3.2 Cyclic and Ideal Lattices

We say a lattice Λ of dimension n is cyclic if for any vector $\boldsymbol{x}$ in Λ, $\mathrm{rot}_{x^n-1}(\boldsymbol{x})$ is also in Λ.

We say a lattice Λ of dimension n is ideal if it is an ideal $I \subseteq R_{\mathbf{f}}$ for some monic and irreducible polynomial $\mathbf{f} \in \mathbb{Z}[x]$ of degree n, that is, there exist $\mathbf{f}$ and $I \subseteq R$ such that $\sigma(I) = \Lambda$. We also say a lattice Λ of dimension n is $\mathbf{f}$-ideal if it corresponds to an ideal $I \subseteq R_{\mathbf{f}}$ under the mapping σ.

Precisely, the lattice $\Lambda(I)$ corresponding I is obtained as follows: Since I is an ideal, there exists a set of polynomial $\mathbf{g}_1, \dots, \mathbf{g}_l$ of degree at most $n-1$ such that $I = \langle \mathbf{g}_1, \dots, \mathbf{g}_l \rangle$. Then, consider $\boldsymbol{G} = [\mathrm{Rot}_{\mathbf{f}}(\mathbf{g}_1)| \dots | \mathrm{Rot}_{\mathbf{f}}(\mathbf{g}_l)]$. The $\Lambda(I)$ is written by $\{\boldsymbol{v} = \boldsymbol{Ge} \in \mathbb{Z}^n \mid \boldsymbol{e} \in \mathbb{Z}^{ln}\}$. By using the standard technique, we have a matrix $\boldsymbol{B} \in \mathbb{Z}^{n\times n}$ such that $\Lambda(I) = \{\boldsymbol{v} = \boldsymbol{Be} \in \mathbb{Z}^n \mid \boldsymbol{e} \in \mathbb{Z}^n\}$.

In addition, we note that any ideal $I \subseteq R_{\mathbf{f}}$ defines the corresponding *full-rank* lattice $\Lambda(I) \subseteq \mathbb{Z}^n$:

Lemma 3.2.1 (Lemma 3.2 [LM06])**.** *Every ideal* $I \subseteq R_{\mathbf{f}}$*, where* $\mathbf{f} \in \mathbb{Z}[x]$ *is a monic and irreducible polynomial of degree n, is isomorphic to a full-rank lattice in* $\mathbb{Z}^n$*.*

Proof. Let $I = \langle \mathbf{g}_1, \dots, \mathbf{g}_l \rangle$, where $\mathbf{g}_i \neq 0$ and they are of degree at most $n-1$. It is obvious that the polynomials $\mathbf{g}_1, \mathbf{g}_1 x, \dots, \mathbf{g}_1 x^{n-1}$ are linearly independent over $\mathbb{Z}$. We show that the polynomials $\mathbf{g}_1, \mathbf{g}_1 \otimes x, \dots, \mathbf{g}_1 \otimes x^{n-1}$ are also linearly independent over $\mathbb{Z}$. This shows the corresponding lattice $\Lambda(I)$ contains n linearly independent vectors $\sigma(\mathbf{g}_1), \sigma(\mathrm{rot}_{\mathbf{f}}(\mathbf{g}_1)), \dots, \sigma(\mathrm{rot}_{\mathbf{f}}^{n-1}(\mathbf{g}_1))$ and completes the proof.

If the polynomials are linearly dependent, then there exists a non-zero polynomial $\mathbf{a} = (a_0 + a_1 x + \cdots + a_{n-1} x^n - 1) \in \mathbb{Z}[x]$ of degree at most $n-1$ such that $\mathbf{g}_1 \otimes \mathbf{a} = \sum_{i=0}^{n} a_i (\mathbf{g}_1 \otimes x^i) = 0$. Then, $\mathbf{g}_1 \cdot \mathbf{a} = \mathbf{f} \cdot \mathbf{h} \in \langle \mathbf{f} \rangle$ for some polynomial $\mathbf{h} \in \mathbb{Z}[x]$. Since $\mathbf{f}$ is irreducible over $\mathbb{Z}$ and $\mathbb{Z}[x]$ is a unique factorization domain, $\mathbf{f}$ is a prime. Thus, $\mathbf{f} \mid \mathbf{g}$ or $\mathbf{f} \mid \mathbf{a}$. Both of $\mathbf{g}_1$ and $\mathbf{a}$ have degree at most $n-1$, this cannot occur unless $\mathbf{g}_1$ or $\mathbf{a}$ is 0. This completes the proof. □

We note that Ding and Lindner [DL07] gave a polynomial-time algorithm which identifies a given basis is a basis spans a lattice or an ideal lattice by employing the Hermite normal form.

Lemma 3.2.2 (Lemma 1 [DL07])**.** *Let* $\boldsymbol{B} \in \mathbb{Z}^{n\times n}$ *be a basis of a lattice* Λ*. Then* Λ *corresponds to some ideal* $I \subseteq R_{\mathbf{f}}$ *if and only if there exists a matrix* $\boldsymbol{T} \in \mathbb{Z}^{n\times n}$ *such* *that*

$$\boldsymbol{R}_{\mathbf{f}}\boldsymbol{B} = \boldsymbol{B}\boldsymbol{T}.$$

We found the extended definitions of ideal lattices in Peikert and Rosen [PR07] and Buchmann and Lindner [BL09]. Let $\mathbb{K} = \mathbb{Q}(\zeta)$ be a number field of degree n for some algebraic number ζ (or you can consider $\mathbb{K} = \mathbb{Q}[x]/\langle \mathbf{f} \rangle$, there is a monic and irreducible polynomial $\mathbf{f} \in \mathbb{Q}[x]$ of degree n such that $\mathbf{f}(\zeta) = 0$). Let O be an order of $\mathbb{K}$.

In [BL09], Buchmann and Lindner said that a lattice Λ is O-ideal if Λ corresponds to some ideal $I \subseteq O$. This definition equals to the one by Lyubashevsky and Micciancio if $O = \mathbb{Z}(\zeta) \simeq \mathbb{Z}[x]/\langle \mathbf{f} \rangle$ and $\mathbf{f} \in \mathbb{Z}[x]$.

In [PR07], Peikert and Rosen said that a lattice Λ is ideal if it corresponds to an ideal $I \subseteq O_{\mathbb{K}}$ through another embedding (the canonical embedding).

3.3 Problems

First of all, we extend the notation of successive minima. For any ideal I of $R_{\mathbf{f}}$, define $\lambda_i^p(I)$ to be $\lambda_i^p(\Lambda(I))$. In the following, we assume that $\mathbf{f} \in \mathbb{Z}[x]$ is a monic polynomial of degree n.

Definition 3.3.1 ($\mathbf{f}$-SPP$_\gamma^p$)**.** Given an ideal $I \subseteq R_{\mathbf{f}}$, the problem is finding a non-zero polynomial $\mathbf{g} \in I$ such that $\|\mathbf{g}\| \leq \gamma \cdot \lambda_1^p(I)$.

Definition 3.3.2 ($\mathbf{f}$-SVP$_\gamma^p$)**.** Given a basis $\boldsymbol{B}$ of a lattice $\Lambda(I)$, where $I \subseteq R_{\mathbf{f}}$, the problem is finding a non-zero vector $\boldsymbol{v} \in \Lambda(I)$ such that $\|\boldsymbol{v}\| \leq \gamma \cdot \lambda_1^p(\Lambda(I))$.

These two problems essentially equals and the difference is only notation.

Naturally, we can define the version of SIVP as follows:

Definition 3.3.3 ($\mathbf{f}$-SIVP$_\gamma^p$)**.** Given a basis $\boldsymbol{B}$ of a lattice $\Lambda(I)$, where $I \subseteq R_{\mathbf{f}}$, the problem is finding a set of linearly independent vectors $\boldsymbol{S} \subset \Lambda(I)$ such that $\|\boldsymbol{S}\|_p \leq \gamma \cdot \lambda_n^p(\Lambda(I))$.

Lyubashevsky and Micciancio gave the following lemma that states the relation of $\lambda_1^\infty(\Lambda(I))$ and $\lambda_n^\infty(\Lambda(I))$. By this, we have the simple reductions from $\mathbf{f}$-SIVP$_\gamma^\infty$ to $\mathbf{f}$-SVP$_\gamma^\infty$ and from $\mathbf{f}$-SVP$_{E_2 \cdot \gamma}^\infty$ to $\mathbf{f}$-SIVP$_\gamma^\infty$ if $\mathbf{f}$ is irreducible, where $E_2 = \mathrm{EF}_\infty(\mathbf{f}, 2)$.

Lemma 3.3.4 (Lemma 4.2 [LM06])**.** *Assume that* $\mathbf{f}$ *is irreducible. For all ideals* $I \subseteq R_{\mathbf{f}}$*, we have*

$$\lambda_n^\infty(\Lambda(I)) \leq \mathrm{EF}_\infty(\mathbf{f}, 2) \cdot \lambda_1^\infty(\Lambda(I)).$$

Proof. Let $\boldsymbol{g} \in \mathbb{Z}^n$ be a shortest vector of $\Lambda(I)$, that is, $\|\boldsymbol{g}\|_\infty = \lambda_1^\infty(\Lambda(I))$. Then, let us consider $\mathbf{g}, \mathbf{g} \otimes x, \ldots, \mathbf{g} \otimes x^{n-1}$. By Lemma 3.2.1, these polynomials are linearly independent. The maximum degree of $\mathbf{g}x^i$ is $2n - 2$. Hence, $\left\|\mathbf{g}x^i\right\|_{\mathbf{f}} \leq \mathrm{EF}_\infty(\mathbf{f}, 2) \cdot \left\|\mathbf{g} \otimes x^i\right\|_\infty \leq \mathrm{EF}_\infty(\mathbf{f}, 2) \cdot \|\mathbf{g}\|_\infty = \mathrm{EF}_\infty(\mathbf{f}, 2)\lambda_1^\infty(\Lambda(I))$ for all $0 \leq i \leq n - 1$. □

Corollary 3.3.5. *Assume that* $\mathbf{f}$ *is irreducible. There exist reductions from* $\mathbf{f}$-SIVP$_\gamma^\infty$ *to* $\mathbf{f}$-SVP$_\gamma^\infty$ *and from* $\mathbf{f}$-SVP$_{E_2 \cdot \gamma}^\infty$ *to* $\mathbf{f}$-SIVP$_\gamma^\infty$*, where* $E_2 = \mathrm{EF}_\infty(\mathbf{f}, 2)$.

Proof. We describe the reduction algorithms for these two reductions.
(From $\mathbf{f}$-SIVP$^{\infty}_{\gamma}$ to $\mathbf{f}$-SVP$^{\infty}_{\gamma}$) The algorithm, given a basis $\boldsymbol{B}$ of a lattice $\Lambda(I)$, invokes the solver of $\mathbf{f}$-SVP$^{\infty}_{\gamma}$ and obtains a non-zero vector $\boldsymbol{g} \in \Lambda(I)$ such that $\|\boldsymbol{g}\|_{\infty} \leq \gamma \cdot \lambda_1^{\infty}(\Lambda(I))$. Then, it outputs $\boldsymbol{S} = \mathrm{Rot}_{\mathbf{f}}(\boldsymbol{g})$ as the solution of the instance $\boldsymbol{B}$ of $\mathbf{f}$-SIVP$^{\infty}_{\gamma}$. The vectors of $\boldsymbol{S}$ are linearly independent and the norm of $\boldsymbol{S}$ is at most $E_2 \cdot \gamma \cdot \lambda_1^{\infty}(\Lambda(I)) \leq \gamma \cdot \lambda_n^{\infty}(\Lambda(I))$.
(From $\mathbf{f}$-SVP$^{\infty}_{E_2\cdot\gamma}$ to $\mathbf{f}$-SIVP$^{\infty}_{\gamma}$) The algorithm, given a basis $\boldsymbol{B}$ of a lattice $\Lambda(I)$, invokes the solver of $\mathbf{f}$-SIVP$^{\infty}_{E_2\cdot\gamma}$ and obtains a set of linearly independent vectors $\boldsymbol{S} = [\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n] \in \Lambda(I)$ such that $\|\boldsymbol{S}\|_{\infty} \leq \gamma \cdot \lambda_n^{\infty}(\Lambda(I))$. Then, it outputs the one of vectors in $\boldsymbol{S}$ as the solution of the instance $\boldsymbol{B}$ of $\mathbf{f}$-SVP$^{\infty}_{E_2\gamma}$. The norm of $\boldsymbol{S}$ is at most $\gamma \cdot \lambda_n^{\infty}(\Lambda(I)) \leq E_2\gamma \cdot \lambda_n^{\infty}(\Lambda(I))$ and this completes the proof. □

In order to show the average-case/worst-case reductions, we define the internal problem as in Section 2.4.

Definition 3.3.6 ($\mathbf{f}$-IncSPP$_{\gamma}$)**.** Given an ideal $I \subseteq R_{\mathbf{f}}$ and a polynomial $\mathbf{g} \in I$ such that $\|\mathbf{g}\|_{\mathbf{f}} > \gamma\lambda_1^{\infty}(I)$, the problem is finding a polynomial $\mathbf{h} \in I$ such that $\|\mathbf{h}\|_{\mathbf{f}} \neq 0$ and $\|\mathbf{h}\|_{\mathbf{f}} \leq \|\mathbf{g}\|_{\mathbf{f}}/2$.

Definition 3.3.7 ($\mathbf{f}$-IncSVP$^{p}_{\gamma}$)**.** Given a basis $\boldsymbol{B}$ of a lattice $\Lambda(I)$ and a vector $\boldsymbol{g} \in \Lambda(I)$ such that $\|\boldsymbol{g}\|_p > \gamma\lambda_1^{\infty}(\Lambda(I))$, where $I \subseteq R_{\mathbf{f}}$, the problem is finding a non-zero vector $\boldsymbol{h} \in \Lambda(I)$ such that $\|\boldsymbol{h}\|_p \leq \|\boldsymbol{g}\|_p/2$.

Lemma 3.3.8 (Lemma 4.4 [LM06])**.** *There is a polynomial-time reduction from* $\mathbf{f}$-SVP$^{\infty}_{\gamma}$ *to* $\mathbf{f}$-IncSVP$^{\infty}_{\gamma}$.

Note on the Hardness: There were no results on the NP-hardness of the above problems. This is the one of main open problems in this area.

3.4 Average-Case/Worst-Case Reductions

3.4.1 From Small Integer Solution Problems

We first extend the definition of the norm $\|\cdot\|_p$. Let us denote a vector of polynomials in $R_{\mathbf{f}}$ or $R_{\mathbf{f},q}$ by $\check{\boldsymbol{a}}$. For $\check{\boldsymbol{e}} = (\mathbf{e}_1, \ldots, \mathbf{e}_m) \in R_{\mathbf{f}}^m$, we write by $\|\check{\boldsymbol{e}}\|_p$ the l_p norm of $\boldsymbol{e} = \sigma(\mathbf{e}_1) \circ \ldots \circ \sigma(\mathbf{e}_m) \in \mathbb{Z}^{mn}$.

We give the definitions of $\mathbf{f}$-SIS and $\mathbf{f}$-ISIS as analogies of SIS and ISIS.

Definition 3.4.1 ($\mathbf{f}$-SIS$^{p}_{q,m,\beta}$)**.** For a monic polynomial $\mathbf{f} \in \mathbb{Z}[x]$ of degree n, integers $m = m(n)$ and $q = q(n)$, a real $\beta = \beta(n)$, given an m-dimensional row vector $\check{\boldsymbol{a}} = [\mathbf{a}_1, \ldots, \mathbf{a}_m] \in R_{\mathbf{f},q}^m$, the problem is finding a non-zero vector $\check{\boldsymbol{e}} = (\mathbf{e}_1, \ldots, \mathbf{e}_m) \in R_{\mathbf{f}}^m$ such that $\check{\boldsymbol{a}}^T\check{\boldsymbol{e}} = \sum_{i\in[m]} \mathbf{a}_i \otimes \mathbf{e}_i = 0$ in $R_{\mathbf{f},q}$ and $\|\check{\boldsymbol{e}}\|_p \leq \beta$.

Definition 3.4.2 ($\mathbf{f}$-ISIS$^{p}_{q,m,\beta}$)**.** For a monic polynomial $\mathbf{f} \in \mathbb{Z}[x]$ of degree n, integers $m = m(n)$ and $q = q(n)$, a real $\beta = \beta(n)$, given an m-dimensional row vector

$\check{\boldsymbol{a}} = [\mathbf{a}_1, \dots, \mathbf{a}_m] \in R_{\mathbf{f},q}^m$ and a polynomial $\mathbf{u} \in R_{\mathbf{f},q}$, the problem is finding a vector $\check{\boldsymbol{e}} = (\mathbf{e}_1, \dots, \mathbf{e}_m) \in R_{\mathbf{f}}^m$ such that $\check{\boldsymbol{a}} \cdot \check{\boldsymbol{e}} = \sum_{i\in[m]} \mathbf{a}_i \otimes \mathbf{e}_i = \mathbf{u}$ in $R_{\mathbf{f},q}$ and $\|\check{\boldsymbol{e}}\|_p \le \beta$.

The problem $\mathbf{f}$-SIS is finding the short non-zero element in the $R_{\mathbf{f}}$-module $M^{\perp}(\check{\boldsymbol{a}}) = \{\check{\boldsymbol{e}} \in R_{\mathbf{f}}^m \mid \check{\boldsymbol{a}} \cdot \check{\boldsymbol{e}} \equiv 0 \pmod{q}\}$. In addition, $\Lambda_q^{\perp}(\check{\boldsymbol{a}}) = \{\boldsymbol{e} \in \mathbb{Z}^{mn} \mid \mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})\cdot \boldsymbol{e} \equiv \mathbf{0} \pmod{q}\}$ is a lattice because this is additive and discrete subgroup of $\mathbb{Z}^{mn}$. Hence, solving $\mathbf{f}$-SIS on $\check{\boldsymbol{a}}$ is finding a short non-zero vector in the lattice $\Lambda_q^{\perp}(\check{\boldsymbol{a}})$.

Micciancio [Mic07] gave the first average-case/worst-case reductions on cyclic lattices. Lyubashevsky and Micciancio [LM06] gave the average-case/worst-case reduction from the ideal-lattice version of the small integer solution problem to $\mathbf{f}$-$\mathrm{SVP}_\gamma^\infty$. We note that the reduction is to the search problem rather than the gap problem.

Lyubashevsky and Micciancio showed the following theorem in [LM06]. Note that m should be larger than $\log q/\log 2\beta$ to ensure the instance of $\mathbf{f}$-$\mathrm{SIS}_{q,m,\beta}^\infty$ has a solution. Note also that the reduction is similar to that of Micciancio and Regev [MR07] and the underlying problem is now $\mathbf{f}$-$\mathrm{SVP}_\gamma^\infty$ since $\lambda_n^\infty(\Lambda(I)) \le \mathrm{EF}_\infty(\mathbf{f}, 2) \cdot \lambda_1^\infty(\Lambda(I))$ by Lemma 3.3.4.

Theorem 3.4.3 ([LM06])**.** *Assume that* $\mathbf{f}$ *is irreducible. Let* $E_3 = \mathrm{EF}_\infty(\mathbf{f}, 3)$*. Let* $m > \log q/\log 2\beta$ *and* $q > 2E_3\beta m n^{3/2} \log n$*. Then for* $\gamma = 8E_3^2 \beta m n \log^2 n$*, there exists an polynomial-time reduction from the worst case of* $\mathbf{f}$-$\mathrm{SVP}_\gamma^\infty$ *to the average case of* $\mathbf{f}$-$\mathrm{SIS}_{q,m,\beta}^\infty$*.*

Stehlé, Steinfeld, Tanaka, and Xagawa gave a variant of the above theorem to save $\sqrt{m} = O(\sqrt{n})$ factor in the reduction from $\mathrm{SIS}_{q,m,\beta}^\infty$ to $\mathrm{SIS}_{q,m,\beta\sqrt{m}}^2$.

Theorem 3.4.4 ([SSTX09])**.** *Suppose that* $\mathbf{f} \in \mathbb{Z}[x]$ *is a monic and irreducible polynomial of degree* n*. Let* $E_k = \mathrm{EF}_\infty(\mathbf{f}, k)$*. Let* $m = \mathrm{poly}(n)$ *be larger than* $\log q/\log 2\beta$ *and* $q = q(n) = \widetilde{\Omega}(E_3 \beta m^2 n)$*. Then if there exists a polynomial-time (resp. subexponential-time) algorithm solving* $\mathbf{f}$-$\mathrm{SIS}_{q,m,\beta}$ *with probability* $1/\mathrm{poly}(n)$ *(resp.* $2^{-o(n)}$*), then there exists a polynomial-time (resp. subexponential-time) algorithm solving* $\mathbf{f}$-SVP_γ *with* $\gamma = \tilde{O}(E_2^2 \beta m n^{1/2})$ *(resp.* $\gamma = \tilde{O}(E^2 \beta m n)$*).*

The proof is essentially the same as one by Lyubashevsky and Micciancio.

To apply the technique of Gentry et al., we need that $R_{\mathbf{f},q}$ be a principal ideal domain. This idea is due to Peikert and Regev [Pei09a].

3.4.2 From Learning With Errors

We next define the parameterized version of the LWE problem.

Definition 3.4.5 ($\mathbf{f}$-$\mathrm{sLWE}_{m,q,\chi}$, the average case)**.** Let χ be a distribution over $\mathbb{T}$. Given $\check{\boldsymbol{a}} \in R_{\mathbf{f},q}^m$ and $\frac{1}{q}\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})^T \boldsymbol{s} + \boldsymbol{x} \in \mathbb{T}^{mn}$, where $\check{\boldsymbol{a}} \leftarrow R_{\mathbf{f},q}^m$ $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{x} \leftarrow \chi^{mn}$, the problem is finding $\boldsymbol{s} \in \mathbb{Z}_q^n$.

You can consider the problem as $\mathbf{f}$-analogue of $\mathrm{sLWE}_{q,\chi}$ with m samples.

For simplicity, we denote $\Lambda_q(\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}}))$ by $\Lambda_q(\check{\boldsymbol{a}})$ if $\mathbf{f}$ is apparent in the context.

Stehlé, Steinfeld, Tanaka, and Xagawa [SSTX09] showed the following theorem.

Theorem 3.4.6 (Theorem 3 [SSTX09])**.** *If there exists an algorithm solving* $\mathrm{sLWE}_{q,m,\Psi_\alpha}$ *in time T and with probability $\epsilon \geq 4m\exp(-\pi/4\alpha^2)$, then there exists a* quantum *algorithm that solves* $\mathrm{SIS}_{q,m,\sqrt{m}/2\alpha}$ *in time* $\mathrm{poly}(T,n)$ *and with probability* $\epsilon^3/64 - O(\epsilon^5) - 2^{-\Omega(n)}$*. The result still holds when replace* sLWE *with* $\mathbf{f}$-sLWE *and* SIS *with* $\mathbf{f}$-SIS *for* $\mathbf{f} = x^n + 1$ *with* $n = 2^k \geq 32$*,* $m \geq 41 \log q$*, and* $q \equiv 3 \pmod 8$*.*

The proof is due to Stehlé and Steinfeld [SS09]. The reduction consists of two reductions, from sLWE to the variant of BDD and from the variant to SIS. The former requires only classical reductions, however, the latter is a quantum reduction.

The variant of BDD is defined as follows:

Definition 3.4.7 (Bounded Distance Decoding with χ, [SSTX09, Definition 3])**.** For a distribution χ, the problem $\mathrm{BDD}(\chi)$ is defined as follows: Given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ and a vector $\boldsymbol{t} = \boldsymbol{b} + \boldsymbol{e}$ where $\boldsymbol{b} \in \Lambda$ and $\boldsymbol{e} \leftarrow \chi$. The goal is to find $\boldsymbol{b}$.

We say that a randomized algorithm $\mathcal{A}$ solves $\mathrm{BDD}(\chi)$ for a lattice Λ with success probability ϵ if, for every $\boldsymbol{b} \in \Lambda$,

$$\Pr_{\boldsymbol{e}\leftarrow\chi,\mathcal{A}}[\mathcal{A}(\boldsymbol{B}, \boldsymbol{t} = \boldsymbol{b} + \boldsymbol{e}) = \boldsymbol{b}] \geq \epsilon.$$

In addition, a randomized algorithm $\mathcal{A}$ solving $\mathrm{BDD}(\chi)$ for a lattice Λ is said to be *strongly solution-independent* if, for every fixed error vector $\boldsymbol{e}$, the probability

$$\Pr_{\mathcal{A}}[\mathcal{A}(\boldsymbol{B}, \boldsymbol{t} = \boldsymbol{b} + \boldsymbol{e}) = \boldsymbol{b}]$$

is independent of $\boldsymbol{b}$.

The first part of the reduction is formally stated as follows:

Lemma 3.4.8 ([SSTX09, Lemma 7])**.** *Let n, q, and m be integers and $\alpha \in (0,1)$ with m and $\log q$ are polynomially bounded by n. Suppose that there exists an algorithm $\mathcal{A}$ that solves* $\mathrm{sLWE}_{m,q,\Psi_{q\alpha}}$*, in time T, and with probability $\epsilon \geq 4m\exp(-\pi/4\alpha^2)$. Then there exists $\mathcal{S} \subseteq \mathbb{Z}_q^{n\times m}$ of cardinality $\epsilon q^{nm}/2$ and an strongly solution-independent algorithm $\mathcal{B}$ such that if $\boldsymbol{A} \in \mathcal{S}$, the algorithm $\mathcal{B}$ solves* $\mathrm{BDD}(\nu_{q\alpha})$ *for $\Lambda_q(\boldsymbol{A})$ in time $T + \mathrm{poly}(n)$ and with probability at least $\epsilon/4$.*

The above algorithm $\mathcal{B}$ is used to construct $\mathrm{BDD}(D_{\Lambda,s})$.

Lemma 3.4.9 ([SSTX09, Lemma 8])**.** *Let $s > 0$ and $\boldsymbol{B}$ be a basis of an n-dimensional lattice Λ. Suppose that there exists a strongly solution-independent algorithm $\mathcal{A}$ that solves* $\mathrm{BDD}(\nu_s)$ *for Λ in time T and with probability ϵ. Then, there exists an integer R such that $|R| = \mathrm{poly}(T, n, |\log s|, |\boldsymbol{B}|)$ and a strongly solution-independent algorithm $\mathcal{B}$ that solves* $\mathrm{BDD}(D_{\Lambda/R,s})$ *within a polynomial time in $\log R$ and with probability at least $\epsilon - 2^{-\Omega(n)}$.*

Regev's quantum reduction is the worst-case/worst-case reduction; That is, if there exists an algorithm $\mathcal{A}$ that solves BDD in the worst case, then there exists a sampler for $D_{\Lambda^*,s}$. Stehlé and Steinfeld observed that the reduction still works even if the algorithm $\mathcal{A}$ only solves the average case.

Lemma 3.4.10 ([SSTX09, Lemma 9])**.** *Suppose we are given a basis $\boldsymbol{B}$ of an n-dimensional lattice Λ, an integer $R > 2^{2n}\lambda_n(\Lambda)$, and a real $s < \lambda_1(\Lambda)/\sqrt{8n}$. Suppose that there exists a strongly solution-independent algorithm $\mathcal{A}$ that solves $\mathrm{BDD}(D_{\Lambda/R,s})$ for Λ with time T and success probability ϵ. Then there exists a quantum algorithm $\mathcal{B}$ which outputs a vector $\boldsymbol{b} \in \Lambda^*$ whose distribution is within distance $1 - \epsilon^2/2 + O(\epsilon^4) + 2^{-\Omega(n)}$ of $D_{\Lambda^*,1/2s}$. Its run-time is $\mathrm{poly}(T, \log R)$.*

We omit the proof and the details, since it deeply relates to quantum computations. Anyway, combining these lemmas, we obtain Theorem 3.4.6.

4 Hash Functions

In this chapter, we give descriptions of one-way and collision-resistant hash functions based on lattice and ideal lattice problems.

Organization: Section 4.1 defines model and security notions on hash schemes. Section 4.2 gives a review on properties of hash functions. Section 4.3 reviews the construction of lattice-based hash functions. We also give the review of ideal-lattice-based hash functions in Section 4.4.

4.1 Definitions

We first give the functional model of a family of hash functions. Let $\mathcal{H}_n = \{h_k : D_n \to R_n \mid k \in K_n\}$ be a *family of hash functions* with the security parameter n, a message space D_n, a digest space R_n, and a key space K_n. Define $\mathcal{H} = \{\mathcal{H}_n\}_n$. We call $\mathcal{H}$ a *hash family* instead of a family of families of hash functions. (Recall the SHA2 family including SHA-224, SHA-256, SHA-384, and SHA-512.)

4.1.1 Model of Hash Schemes

A (cryptographic) hash scheme Hash is a pair of algorithms $(\mathsf{Setup}, \mathsf{Eval})$.

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs a key k.

$\mathsf{Eval}(k, msg)$**:** An evaluation algorithm, given k and a message $msg \in D_n$, returns a digest $d \in R_n$.

We define $h_k(msg) = \mathsf{Eval}(k, msg)$. By this definition, we can identify a hash scheme Hash with a hash family $\mathcal{H}$.

4.1.2 Security Notions

Roughly speaking, we say that Hash is one-way if any polynomial-time adversary cannot, given k and a random image u, output a preimage of y under the hash function indexed by k. We say that Hash is collision resistant if any polynomial-time adversary cannot, given k, output a collision (msg, msg') of the hash function indexed by k. Formally, we define the following experiments $\mathbf{Exp}^{\mathrm{ow}}_{\mathsf{Hash},\mathcal{A}}(n)$ and $\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{Hash},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$ for the one-way and the collision-resistant properties of a hash scheme.

Experiment $\mathbf{Exp}^{\mathrm{ow}}_{\mathsf{Hash},\mathcal{A}}(n)$:

Setup Phase: The challenger C runs Setup with 1^n and obtains k. Next, C generates a random element $msg \leftarrow D_n$ and computes $u \leftarrow \mathsf{Eval}(k, msg)$. C feeds k and msg to the adversary $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ outputs msg'. If $msg' \in D_n$ and $\mathsf{Eval}(k, msg') = u$ then the challenger returns 1, otherwise, 0.

Experiment $\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{Hash},\mathcal{A}}(n)$:

Setup Phase: The challenger C runs Setup with 1^n and obtains k. C feeds k to the adversary $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ outputs msg and msg'. If $msg, msg' \in D_n$, $msg \neq msg'$, and $\mathsf{Eval}(k, msg) = \mathsf{Eval}(k, msg')$ then the challenger returns 1, otherwise, 0.

Definition 4.1.1. Let $\mathsf{Hash} = (\mathsf{Setup}, \mathsf{Eval})$ be a hash scheme. Let $\mathcal{A}$ be an adversary. Let the advantage of $\mathcal{A}$ against one-way property be $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{Hash},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{ow}}_{\mathsf{Hash},\mathcal{A}}(n) = 1\right]$. We say that Hash is one-way if, for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{Hash},\mathcal{A}}(n)$ is negligible in n.

Let the advantage of $\mathcal{A}$ against collision-resistance property be $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{Hash},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{Hash},\mathcal{A}}(n) = 1\right]$. We say that Hash is collision resistant if, for any probabilistic polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{Hash},\mathcal{A}}(n)$ is negligible in n.

There are several security notions on (cryptographic) hash schemes: one-wayness (first-preimage resistance), second-preimage resistance, etc. On the definitions of them and the relations between them, see [RS04].

4.2 Probabilistic Notions on Hash Functions and the Leftover Hash Lemma

In addition to the above security notions, we often discuss other notions on hash families in this thesis and the leftover hash lemma.

First of all, we recall the probabilistic notions on a family of hash functions (see Shoup's textbook [Sho08, Section 8.7]). Again, let $\mathcal{H}_n = \{h_k : D_n \to R_n \mid k \in K_n\}$

be a family of hash functions with the security parameter n, a message space D_n, a digest space R_n, and a key space K_n.

- We say $\mathcal{H}_n$ is *ϵ-almost universal* if for all $x \neq x' \in D_n$,

$$\Pr_{k \leftarrow K_n}[h_k(x) = h_k(x')] \leq \epsilon.$$

- We also say that $\mathcal{H}_n$ is *universal* if it is $1/\left|R_n\right|$-almost universal.
- We say $\mathcal{H}_n$ is *ϵ-almost strongly universal* if $h_k(x)$ is uniformly distributed over R_n, that is $\Pr_{h \leftarrow K_n}[h_k(x) = y] = 1/\left|R_n\right|$ for any $x \in D_n$ and $y \in R_n$, *and* for all distinct $x, x' \in D_n$ and for all $y, y' \in R_n$,

$$\Pr_{k \leftarrow K_n}[h_k(x) = y \wedge h_k(x') = y'] = \frac{\epsilon}{|R_n|}.$$

- We also say that $\mathcal{H}_n$ is *pairwise independent* if it is $1/\left|R_n\right|$-almost strongly universal.

We naturally extend these notions of a family of hash functions to a hash family $\mathcal{H} = \{\mathcal{H}_n\}_n$.

The Leftover Hash Lemma: The leftover hash lemma appears anywhere of areas in the computer science and cryptography.

We follow the presentation by Shoup [Sho08]. Let $|D_n| = N$ and $|R_n| = M$. Let K be a random variable uniformly distributed over K_n and let X be a random variable distributed over D_n. The collision probability of X is defined as $\beta = \sum_{x \in D_n} \Pr[X = x]^2$. The quantity $\log(1/\gamma)$ is the min entropy of X (see Section 1.2).

The following versions are somewhat generalized versions of the leftover hash lemma.

Lemma 4.2.1 (Thm.8.37, [Sho08]). *Let $\mathcal{H}_n$ be a $(1+\alpha)/M$-almost universal family of hash functions from D_n to R_n. Then,*

$$\Delta((K, h_K(X)), (K, U)) \leq \frac{1}{2}\sqrt{M\beta + \alpha}.$$

Lemma 4.2.2 (Thm.8.38, [Sho08]). *Let $\mathcal{H}_n$ be a $(1+\alpha)/M$-almost universal family of hash functions from D_n to R_n. Then,*

$$\Delta((K, h_K(X_1), \ldots, h_K(X_l)), (K, U_1, \ldots, U_l)) \leq \frac{1}{2} l \sqrt{M\beta + \alpha}.$$

Lemma 4.2.3 (Leftover Hash Lemma (a min-entropy version)). *Let $\mathcal{H}_n = \{h_k : D_n \to R_n \mid k \in K_n\}$ be a family of hash functions, where D_n and R_n are finite sets. Let K be the uniform distribution over K_n. and let X be a random variable distributed according to D. Then,*

$$\Delta((K, h_K(X)), (K, U)) \leq 2^{-\frac{1}{2}\left(H_\infty(X) - \log|R_n| + 2\right)}$$

where U is a random variable distributed uniformly over R_n. In particular, if X is distributed uniformly over D_n, we have

$$\Delta((K, h_K(X)), (K, U)) \le \frac{1}{2}\sqrt{\frac{|R_n|}{|D_n|}}.$$

4.3 Lattice-based hash functions

We review the lattice-based hash functions. For two integers $q = q(n)$ and $m = m(n)$, we define a family of hash functions,

$$\mathcal{H}_n(q, m) = \{h_{\boldsymbol{A}} : D_n \to \mathbb{Z}_q^n \mid \boldsymbol{A} \in \mathbb{Z}_q^{n\times m}\},$$

where $h_{\boldsymbol{A}}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} \bmod q$ and $D_n \subseteq \mathbb{Z}^m$.

The definition of the hash scheme is as follows:

Scheme 4.3.1 (LHash)**.** This scheme is parametrized by integers $m = m(n)$ and $q = q(n)$, and a space $D_n \subseteq \mathbb{Z}^m$. The key space is $\mathbb{Z}_q^{n\times m}$. The message space is D_n and the digest space is $R_n = \mathbb{Z}_q^n$.

Setup(1^n)**:** Given 1^n, output $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$.

Eval($\boldsymbol{A}, \boldsymbol{e}$)**:** Given $\boldsymbol{A}$ and $\boldsymbol{e} \in D_n$, output $\boldsymbol{A}\boldsymbol{e} \bmod q$.

We can identify $\mathcal{H}(q, m) = \{\mathcal{H}_n(q, m)\}$ with LHash = (Setup, Eval).

It is easily verified that the collision resistance and the one-wayness is directly connected to the average-case hardness of $\mathrm{SIS}^p_{q,m,2\beta}$ and $\mathrm{ISIS}^p_{q,m,\beta}$, respectively, where β is the upperbound of the l_p norm of $\boldsymbol{x} \in D_n$. If we set $D_n = \{0, 1\}^m$, the underlying problem is $\mathrm{SIS}_{q,m,\sqrt{m}}$. Hence, as we review in Section 2.4.2, the hash scheme is collision resistance if GapSVP_γ or SIVP_γ is hard in the worst case. Below we give the brief history and the precise security on this LHash.

Originally, Ajtai [Ajt96] showed that the worst-case hardness of GapSVP_γ for some polynomial $\gamma(n)$ is reduced to the average-case hardness of $\mathrm{SIS}_{q,m,n}$ for suitable $q(n)$ and $m(n)$. It is known that $\mathcal{H}(q, m)$ is indeed collision resistant for suitably chosen q and m by Goldreich, Goldwasser, and Halevi [GGH96]. They observed that finding a collision $(\boldsymbol{x}, \boldsymbol{x}')$ for $h_{\boldsymbol{A}} \in \mathcal{H}(q, m)$ implies finding a short non-zero vector $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}'$ such that $\|z\| \le \sqrt{m}$ and $\boldsymbol{A}\boldsymbol{z} \equiv \boldsymbol{0} \pmod q$, i.e., solving $\mathrm{SIS}_{q,m,\sqrt{m}}$. Cai and Nerurkar [CN97] and Micciancio [Mic04] improved an approximation factor of the underlying lattice problems, where $\gamma = \tilde{O}(n^4)$ and $\tilde{O}(n^3)$, respectively. Micciancio and Regev showed that $\mathcal{H}(q, m)$ is collision resistant under the assumption that $\mathrm{GapSVP}_{\tilde{O}(n)}$ is hard in the worst case [MR07], which is a drastic improvement. There were another reductions from the gap version of the covering radius problem GapCRP_γ, the shortest independent vector problem SIVP_γ, and the guaranteed distance decoding problem GDD_γ by adjusting the parameters [MR07]. Following it, Peikert [Pei08] showed the reductions from the same problems in any l_p norms for $p \ge 2$. A recent paper [GPV08, Section 9] by Gentry, Peikert, and

Vaikuntanathan showed that the modulus q in SIS can be $\tilde{O}(n)$ as we already noted in Section 2.4.2.

Theorem 4.3.2 (Implicit in [GPV08]). *Let $m = m(n)$, $q = q(n)$, $\beta = \beta(n)$, $\gamma = \gamma(n)$ be polynomially-bounded functions. For any $g(n) = \omega(\log n)$, there exists a negligible function $\epsilon = \epsilon(n)$ such that, for any $q \geq \gamma \cdot \omega(\sqrt{\log n})$ and $\gamma = \beta\sqrt{n} \cdot \omega(\sqrt{\log n})$, there is a probabilistic polynomial-time reduction from* SIVP_γ *in the worst case to* $\mathrm{SIS}_{q,m,\beta}$ *or* $\mathrm{ISIS}_{q,m,\beta}$ *on average.*

4.3.1 Regularity

In the literature, Ajtai firstly showed the regularity of the hash function. Regev improved the analysis of the regularity.

Lemma 4.3.3 (Claim 5.3, adapted [Reg09]). *Let $\mathbb{G}$ be some finite Abelian group of cardinality Q and let m be some integer. For any m element $g_1, \dots, g_m$, consider $\Delta(\sum_{i \in [m]} b_i g_i, u)$, where $b_i \leftarrow \{0,1\}$ and $u \leftarrow \mathbb{G}$. Then, the expectation of this statistical distance over a uniform choice of $g_1, \dots, g_m$ is at most $(Q/2^m)^{1/2}$. In particular, the probability that this statistical distance is more than $(Q/2^m)^{1/4}$ is at most $(Q/2^m)^{1/4}$.*

A strategy to obtain the bound on the statistical distance is showing the family of hash functions $\mathcal{H}_{n,\mathbb{G}} = \{h_{\boldsymbol{g}} : \{0,1\}^m \to \mathbb{G} \mid \boldsymbol{g} \in \mathbb{G}^m\}$, where $h_{\boldsymbol{g}}(\boldsymbol{b}) = \sum_{i \in [m]} b_i g_i$, is universal and applying the leftover hash lemma. In [Reg09], Regev essentially showed that the hash is universal. In addition, he also gave the bound of expectation. We review his proof closer.

Proof. For $\boldsymbol{g} = (g_1, \dots, g_m) \in \mathbb{G}^m$, let us define $P_{\boldsymbol{g}}(h) = \frac{1}{2^m}\left|\{\boldsymbol{b} \in \{0,1\}^m \mid \sum_i b_i g_i = h\}\right|$. Notice that for any $\boldsymbol{b} \neq \boldsymbol{b}'$, $\Pr_{\boldsymbol{g} \leftarrow \mathbb{G}^m}[\sum_i b_i g_i = \sum_i b'_i g_i] = \Pr_{\boldsymbol{g} \leftarrow \mathbb{G}^m}[\sum_i (b_i - b'_i) g_i] = 1/Q$, since $\mathbb{G}$ is Abelian. This already showed that the family of hash functions $\mathcal{H}_{n,\mathbb{G}}$ is universal. In particular, we obtain that, by applying the leftover hash lemma,

$$\Delta((\boldsymbol{g}, \textstyle\sum_i b_i g_i), (\boldsymbol{g}, u)) \leq \frac{1}{2}\sqrt{\frac{Q}{2^m}}$$

since the collision probability of $\boldsymbol{b}$ is $1/2^m$.

Next, we bound the collision probability for fixed $\boldsymbol{g} \in \mathbb{G}^m$, that is, the square of the l_2 norm of the function $P_{\boldsymbol{g}}$ over $\mathbb{R}^Q$. We can upper bound this by as follows:

$$\begin{aligned} \sum_{h \in \mathbb{G}} P_{\boldsymbol{g}}(h)^2 &= \Pr_{\boldsymbol{b},\boldsymbol{b}' \leftarrow \{0,1\}^m}\left[\sum_i b_i g_i = \sum_i b'_i g_i\right] \\ &\leq \frac{1}{2^m} + \Pr_{\boldsymbol{b},\boldsymbol{b}' \leftarrow \{0,1\}^m}\left[\sum_i b_i g_i = \sum_i b'_i g_i \mid \boldsymbol{b} \neq \boldsymbol{b}'\right]. \end{aligned}$$

Hence, taking $\boldsymbol{g}$ as a random variable, we obtain that

$$\operatorname*{Exp}_{g \leftarrow \mathbb{G}^m}\left[\sum_{h \in \mathbb{G}} P_{\boldsymbol{g}}(h)^2\right] \in \frac{1}{2^m} \pm \frac{1}{Q}.$$

By using the norm bound $\|\boldsymbol{x}\|_\infty \leq \sqrt{Q}\,\|\boldsymbol{x}\|_2$ for any $\boldsymbol{x} \in \mathbb{R}^Q$, we have that

$$\begin{aligned}
\operatorname*{Exp}_{\boldsymbol{g}}\left[\sum_h \left|P_{\boldsymbol{g}}(h) - \frac{1}{Q}\right|\right] &\leq \operatorname*{Exp}_{\boldsymbol{g}}\left[Q^{1/2}\left(\sum_h \left(P_{\boldsymbol{g}}(h) - \frac{1}{Q}\right)^2\right)^{1/2}\right] \\
&= Q^{1/2} \operatorname*{Exp}_{\boldsymbol{g}}\left[\left(\sum_h \left(P_{\boldsymbol{g}}(h) - \frac{1}{Q}\right)^2\right)^{1/2}\right] \\
&\leq Q^{1/2}\left(\operatorname*{Exp}_{\boldsymbol{g}}\left[\sum_h P_{\boldsymbol{g}}(h)^2\right] - \frac{1}{Q}\right)^{1/2} \\
&\leq Q^{1/2} \cdot 2^{-m/2}.
\end{aligned}$$

Hence,

$$\operatorname*{Exp}_{\boldsymbol{g}}[\Delta(\textstyle\sum_i b_i g_i, u)] \leq \tfrac{1}{2} Q^{1/2} 2^{-m/2}.$$

By using the average argument, we have that

$$\Pr_{\boldsymbol{g}}\left[\Delta(\textstyle\sum_i b_i g_i, u) \geq Q^{1/4} 2^{-m/4}\right] \leq Q^{1/4} 2^{-m/4}.$$

□

Notice that this argument can be applied to any $\mathbb{G} = \mathbb{Z}_q^{n+l}$. In particular, for $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{(n+l)\times m}$, $\boldsymbol{e} \leftarrow \{0,1\}^m$, and $\boldsymbol{u} \leftarrow \mathbb{Z}_q^{n+l}$, we have that

$$\Delta((\boldsymbol{A}, \boldsymbol{A}\boldsymbol{e}), (\boldsymbol{A}, \boldsymbol{u})) \leq 2^{-\frac{1}{2}(m+1-(n+l)\log q)}.$$

Hence, taking $m \geq ((1+\delta)n + l)\log q$ for some constant $\delta > 0$, we obtain the (statistical) regularity of the lattice-based hash family.

4.4 Ideal-Lattice-Based Hash Functions

For $\mathbf{a}_1, \ldots, \mathbf{a}_m \in R_{\mathbf{f},q}$, let $\check{\boldsymbol{a}}$ represent an m-dimensional row vector $[\mathbf{a}_1, \ldots, \mathbf{a}_m]$.

Let us define a family of hash functions.

$$\mathcal{H}_n(\mathbf{f}, q, m) = \left\{h_{\check{\boldsymbol{a}}} : D_n \to R_{\mathbf{f},q} \simeq \mathbb{Z}_q^n \mid \check{\boldsymbol{a}} \in R_{\mathbf{f},q}^m\right\},$$

where $h_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}}) = \check{\boldsymbol{a}} \cdot \check{\boldsymbol{e}}$ and $\check{\boldsymbol{e}} = (\mathbf{e}_1, \ldots, \mathbf{e}_m) \in D_n$. We define a hash family $\mathcal{H}(\mathbf{f}, q, m) = \{\mathcal{H}_n(\mathbf{f}, q, m)\}_n$. We note that this hash function is a special version of the lattice-based hash functions. To confirm this, verify the following relations: Let $\boldsymbol{e} = \boldsymbol{e}_1 \circ \ldots \circ \boldsymbol{e}_m$ for $\boldsymbol{e}_m \in \mathbb{Z}^n$. Then, we identify $\boldsymbol{e}$ with $\check{\boldsymbol{e}} = (\mathbf{e}_1, \ldots, \mathbf{e}_m) \in R_{\mathbf{f}}^m$. So, we have that $\check{\boldsymbol{a}} \cdot \check{\boldsymbol{e}} = \mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})\boldsymbol{e}$.

Scheme 4.4.1 (ILHash)**.** The hash scheme ILHash = (Setup, Eval) is parametrized by monic polynomials $\mathbf{f} \in \mathbb{Z}[x]$ of degree n, integers $m = m(n)$ and $q = q(n)$, and a space $D_n \subseteq R_{\mathbf{f}}^m \simeq \mathbb{Z}^{mn}$. The key space is $R_{\mathbf{f},q}^m$. The message space is D_n and the digest space is $R_n = R_{\mathbf{f},q} \simeq \mathbb{Z}_q^n$.

Setup(1^n)**:** Given 1^n, output $\check{\boldsymbol{a}} \leftarrow R_{\mathbf{f},q}^m$.

Eval($\check{\boldsymbol{a}}, \check{\boldsymbol{e}}$)**:** Given $\check{\boldsymbol{a}}$ and $\check{\boldsymbol{e}} \in D_n$, output $\check{\boldsymbol{a}} \cdot \check{\boldsymbol{e}} \bmod q$.

The first compact hash function is by Micciancio [Mic07] and with parameters $\mathbf{f} = x^n - 1$ and $D_n = ([D]^n)^m$ for a small integer D. He proved that this hash functions are one-way if $(x^n - 1)$-$\mathrm{SVP}_\gamma^\infty$ is hard in the worst-case for certain γ and parameter settings. He left the open problem deciding whether this functions are collision resistant or not.

This problem is solved negatively by Peikert and Rosen [PR06] and Lyubashevsky and Micciancio [LM06] with demonstration of the attacks finding the collision. The polynomial $x^n - 1$ has the small factor $x - 1$ over $\mathbb{Z}$ and, thus, over $\mathbb{Z}_q$. Hence, $\mathbf{a}_1$ is divisible by $x - 1$ with probability $1/q$ over the choice of $\mathbf{a}_1$. Suppose that happens. Then, we set $\mathbf{z}_1 = (x^n - 1)/(x - 1) = x^{n-1} + x^{n-2} + \cdots + x + 1 \in D^n$ and $\mathbf{z}_i = 0$ for $i = 2, \ldots, m$. Obviously, we have that $\mathbf{a}_1 \otimes \mathbf{z}_1 = 0$ even over $\mathbb{Z}$. The pair $(\check{\boldsymbol{e}}, \check{\boldsymbol{e}}') = ((\mathbf{z}_1, 0, \ldots, 0), (0, 0, \ldots, 0))$ is collision if $\mathbf{a}_1$ is divisible by $x - 1$.

The point is that the ring $\mathbb{Z}[x]/\langle x^n - 1\rangle$ is not an integral domain.[1] To fix the weak point, Peikert and Rosen [PR06], and Lyubashevsky and Micciancio [LM06] proposed the technique, use of an integral domain. Peikert and Rosen gives an algebraic constraint to D_n to avoid the weak point. Lyubashevsky and Micciancio set the polynomial $\mathbf{f}$ to be irreducible over $\mathbb{Z}$, in order to ensure $R_{\mathbf{f}} = \mathbb{Z}[x]/\langle \mathbf{f}\rangle$ an integral domain, (hence, $\mathbb{Q}[x]/\langle \mathbf{f}\rangle$ is a field). Their techniques are essentially same. We adopt the latter for generalization.

Applying Theorem 3.4.3 by Lyubashevsky and Micciancio [LM06], we obtain the following security theorem.

Theorem 4.4.2 ([LM06])**.** *Let* $\mathbf{f} \in \mathbb{Z}[x]$ *be a monic and irreducible polynomial of degree n and let* $E_3 = \mathrm{EF}_\infty(\mathbf{f}, 3)$*. Let* β *be the upperbound of the* l_∞ *norm of vectors in* D_n*. Let* $m > \log q / \log 2\beta$ *and* $q > 2E_3\beta m n^{3/2} \log n$*. Then for* $\gamma = 8E_3^2 mn \log^2 n$*, if* $\mathbf{f}$-$\mathrm{SVP}_\gamma^\infty$ *is hard in the worst case then* ILHash *is collision resistant.*

4.4.1 Computational Tricks

Hereafter, we describe why ILHash is attractive on computational issues (see also the original papers [Mic07, PR06, LM06, LMPR08]).

Notice that we can set $m = \Theta(\log n)$ and $q = \mathrm{poly}(n)$ in the above ILHash. Hence, ILHash enjoys the compactness of the parameter $\check{\boldsymbol{a}} \in R_{\mathbf{f},q}^m$ rather than $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. The computation of ILHash.Eval is also reduced to $\tilde{O}(n)$ by a careful choice of the parameters.

[1] We say that a ring R is an integral domain if $ab \neq 0$ for any non-zero elements $a, b \in R$.

Given a parameter $\breve{\boldsymbol{a}} = (\mathbf{a}_1, \ldots, \mathbf{a}_m)$ and a message $\breve{\boldsymbol{e}} = (\mathbf{e}_1, \ldots, \mathbf{e}_m)$, the hash value is $\sum_{i \in [m]} \mathbf{a}_i \otimes \mathbf{e}_i$. Since m can be set as $\Theta(\log n) = \tilde{O}(1)$, it is suffice to show that the multiplication in $R_{\mathbf{f},q} = \mathbb{Z}_q[x]/\langle \mathbf{f} \rangle$ costs only $\tilde{O}(n)$ if we take $\mathbf{f}$ and q carefully.

The Fourier transformation: First, assume that $\mathbf{f}$ splits $\mathbb{Z}_q$ completely, that is, $\mathbf{f}(x) = \prod_{i \in [n]}(x - w_i)$ over $\mathbb{Z}_q$. In this case we can use, to compute $\mathbf{a} \otimes \mathbf{e}$ in $R_{\mathbf{f},q}$, the discrete Fourier transformation (or the number theoretic transformation) which is an isomorphism from $R_{\mathbf{f},q}$ to $\mathbb{Z}_q^n$. By using the Chinese reminder theorem, we have that

$$R_{\mathbf{f},q} = \mathbb{Z}_q[x]/\langle \mathbf{f} \rangle \simeq \prod_{i \in [n]} \mathbb{Z}_q[x]/\langle x - w_i \rangle,$$

where the isomorphism ϕ is given by $\phi(\mathbf{a}) = (\mathbf{a} \bmod x - w_1, \ldots, \mathbf{a} \bmod x - w_n)$. Notice that $\mathbf{a}(w_1) = \mathbf{a} \bmod x - w_1$. Henceforth, $\prod_{i \in [n]} \mathbb{Z}_q[x]/\langle x - w_i \rangle \simeq \mathbb{Z}_q^n$, where addition and multiplication is defined by the pairwise ones. We rename ϕ by DFT in the following and write $\hat{\mathbf{a}} = \mathrm{DFT}(\mathbf{a})$ for any polynomial $\mathbf{a} \in R_{\mathbf{f},q}$. Define the matrix $\boldsymbol{W}_{\mathrm{DFT}}$ by $\{w_i^{j-1}\}_{i,j \in [n]}$. Notice that

$$\mathrm{DFT}(\mathbf{a}) = \boldsymbol{W}_{\mathrm{DFT}} \cdot \boldsymbol{a} = \begin{pmatrix} w_1^0 & w_1^1 & \ldots & w_1^n \\ w_2^0 & w_2^1 & \ldots & w_2^n \\ \vdots & \vdots & \ddots & \vdots \\ w_n^0 & w_n^1 & \ldots & w_n^n \end{pmatrix} \cdot \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} \mathbf{a}(w_1) \\ \mathbf{a}(w_2) \\ \vdots \\ \mathbf{a}(w_n) \end{pmatrix}.$$

To compute $\mathbf{a} \otimes \mathbf{e}$ in $R_{\mathbf{f},q}$, we compute as follows: (1) compute $\hat{\mathbf{a}} = \mathrm{DFT}(\mathbf{a}) = (\mathbf{a}(w_1), \ldots, \mathbf{a}(w_n))$ and $\hat{\mathbf{e}} = \mathrm{DFT}(\mathbf{e})(\mathbf{e}(w_1), \ldots, \mathbf{e}(w_n))$, (2) compute $\widehat{\mathbf{a} \otimes \mathbf{e}} = \hat{\mathbf{a}} \cdot \hat{\mathbf{e}} = (\mathbf{a}(w_1) \cdot \mathbf{e}(w_1), \ldots, \mathbf{a}(w_1) \cdot \mathbf{e}(w_n))$, and (3) we obtain $\mathbf{a} \otimes \mathbf{e}$ by applying DFT^{-1}.

To fasten the convolutions $\mathbf{a} \otimes \mathbf{e}$, we possibly take $\mathbf{f} = x^{2^k} - 1$ and a prime q such that $2^k \mid q - 1$. By this choice, there is an element $w \in \mathbb{Z}_q^*$ such that the cardinality of the subgroup generated by w is 2^k. Then, $\mathbf{f}(x) = \prod_{i \in [n]}(x - w^{i-1})$ and $\boldsymbol{W}_{\mathrm{DFT}} = \{w^{(i-1)(j-1)}\}_{i,j \in [n]}$. This enables us to use of the fast Fourier transform (FFT).

The another choice is $\mathbf{f} = x^{2^k} + 1$ and a prime q such that $2^{k+1} \mid q - 1$. In this choice, we have an element $w \in \mathbb{Z}_q^*$ such that the cardinality of the subgroup generated by w is 2^{k+1}. Then, $\mathbf{f}(x) = \prod_{i \in [n]}(x - w^{2i+1})$ and $\boldsymbol{W}_{\mathrm{DFT}} = \{w^{(2i+1)(j-1)}\}_{i,j \in [n]}$. Despite of some differences, again, this matrix $\boldsymbol{W}_{\mathrm{DFT}}$ allows us to use of the techniques in the FFT. See the below.

The Fast Fourier Transform (FFT) over $\mathbb{Z}_q[x]/\langle x^{2^k} + 1 \rangle$: It is well-known that $\mathrm{DFT}(\mathbf{f})$ can be computed by $O(n \log n)$ additions and multiplications. Let us define

$\boldsymbol{W}_{n,w} = \{w^{(i-1)(j-1)}\}_{i,j\in[n]}$. From the definition of $\boldsymbol{W}_{n,w}$, in this case we have hat

$$\mathrm{DFT}(\mathbf{a}) = \boldsymbol{W}_{n,w} \cdot \boldsymbol{a} = \begin{pmatrix} w^0 & w^1 & w^2 & \cdots & w^{n-1} \\ w^0 & w^3 & w^6 & \cdots & w^{3(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w^0 & w^{2n-1} & w^{4(n-1)} & \cdots & w^{(n-1)(n-1)} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{pmatrix}.$$

Let us consider the simple case that $n = 4$. In the case, the order of w is 8 and we have that

$$\boldsymbol{W}_{4,w} \cdot \boldsymbol{a} = \begin{pmatrix} w^0 & w^1 & w^2 & w^3 \\ w^0 & w^3 & w^6 & w^1 \\ w^0 & w^5 & w^2 & w^7 \\ w^0 & w^7 & w^6 & w^5 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}.$$

Swapping the right hand side by the permutation $\pi = (0, 2, 1, 3)$ over $\{0, 1, 2, 3\}$,

$$\boldsymbol{W}_{4,w} \cdot \boldsymbol{a} = \begin{pmatrix} w^0 & w^2 & w^1 & w^3 \\ w^0 & w^6 & w^3 & w^1 \\ w^0 & w^2 & w^5 & w^7 \\ w^0 & w^6 & w^7 & w^5 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_2 \\ a_1 \\ a_3 \end{pmatrix} = \begin{pmatrix} w^{2\cdot 0} & w^{2\cdot 1} & w^1 \cdot w^{2\cdot 0} & w^1 \cdot w^{2\cdot 1} \\ w^{2\cdot 0} & w^{2\cdot 3} & w^3 \cdot w^{2\cdot 0} & w^3 \cdot w^{2\cdot 3} \\ w^{2\cdot 0} & w^{2\cdot 1} & w^5 \cdot w^{2\cdot 0} & w^5 \cdot w^{2\cdot 1} \\ w^{2\cdot 0} & w^{2\cdot 3} & w^7 \cdot w^{2\cdot 0} & w^7 \cdot w^{2\cdot 3} \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_2 \\ a_1 \\ a_3 \end{pmatrix}.$$

Since the order of w is 8, then $w^4 = -1$ in $\mathbb{Z}_q$. Hence, we have that

$$\boldsymbol{W}_{4,w} \cdot \boldsymbol{a} = \begin{pmatrix} \boldsymbol{W}_{2,w^2} \cdot \boldsymbol{a}_e + (w, w^3) \odot (\boldsymbol{W}_{2,w^2} \cdot \boldsymbol{a}_o) \\ \boldsymbol{W}_{2,w^2} \cdot \boldsymbol{a}_e - (w, w^3) \odot (\boldsymbol{W}_{2,w^2} \cdot \boldsymbol{a}_o) \end{pmatrix},$$

where $\boldsymbol{a}_e = (a_0, a_2)$, $\boldsymbol{a}_o = (a_1, a_3)$, and $\odot$ denotes the pairwise multiplication in $\mathbb{Z}_q^2$.

This holds also any $n = 2^k$ and w a generator of a 2^{k+1} subgroup of $\mathbb{Z}_q^*$. Generally we have that

$$\boldsymbol{W}_{n,w} \cdot \boldsymbol{a} = \begin{pmatrix} \boldsymbol{W}_{n/2,w^2} \cdot \boldsymbol{a}_e + (w, w^3, \dots, w^{2n-1}) \odot (\boldsymbol{W}_{n/2,w^2} \cdot \boldsymbol{a}_o) \\ \boldsymbol{W}_{n/2,w^2} \cdot \boldsymbol{a}_e - (w, w^3, \dots, w^{2n-1}) \odot (\boldsymbol{W}_{n/2,w^2} \cdot \boldsymbol{a}_o) \end{pmatrix}.$$

Computing recursively, we can obtain $\mathrm{DFT}(\mathbf{a}) = \boldsymbol{W}_{n,w} \cdot \boldsymbol{a}$ with $O(n \log n)$ additions and multiplications in $\mathbb{Z}_q$ if we precompute $(w, w^2, w^3, \dots, w^{2n})$. See [LMPR08] for the implementation issues.

Choices of a polynomial f: We can use the FFT even if $\mathbf{f}$ is not split $\mathbb{Z}_q$ completely. The idea is embedding $R_{\mathbf{f},q}$ into another $R_{\mathbf{f}',q'}$. Let $\mathbf{f}' = x^{n'} - 1$ with $n' = 2^{k'} > 2(n-1)$ and q' be a prime such that $q' > nq^2$ and $n' \mid q' - 1$. Then, $R_{\mathbf{f}',q'}$ admits the fast Fourier transformation. For $\mathbf{a}, \mathbf{b} \in R_{\mathbf{f},q}$, consider $\mathbf{a} \cdot \mathbf{b}$ in $\mathbb{Z}[x]$. By the hypothesis on $\mathbf{f}'$ and q', $\mathbf{a} \cdot \mathbf{b}$ equals to $\mathbf{a} \otimes \mathbf{b}$ in $R_{\mathbf{f}',q'}$. Hence, we first compute $\mathbf{a} \otimes \mathbf{b}$ in $R_{\mathbf{f}',q'}$ and reduce it modulo $\mathbf{f}$ and q.

If $\mathbf{f}$ is $x^{2^k} \pm 1$ but not split $\mathbb{Z}_q$ completely, we have no need to choose another $\mathbf{f}'$. We embed $R_{\mathbf{f},q}$ into $R_{\mathbf{f},q'}$ which admits the fast Fourier transformation (to do so, $q' > nq^2$ is a prime such that $2^k \mid q' - 1$ or $2^{k+1} \mid q' - 1$).

4.4.2 Micciancio's Regularity Lemma

In this section, we examine the regularity of the hash family $\mathcal{H}(\mathbf{f}, q, m)$.

Micciancio proved the regularity of the hash family $\mathcal{H}(x^n - 1, q, m)$ with a message space $D_n = [D]^{mn}$. The proof can be applied to the hash family $\mathcal{H}(\mathbf{f}, q, m)$ with closer look of the proof in [Mic07].

Lemma 4.4.3 (Lemma 6 in [SSTX09], adapted version of Theorem 4.2 in [Mic07])**.** *Let $\mathbb{F}$ be a finite field and $\mathbf{f} \in \mathbb{F}[x]$ be monic and of degree n. Let R be the ring $\mathbb{F}[x]/\langle \mathbf{f} \rangle$. Let $\mathcal{D} \subseteq \mathbb{F}$. For a row vector of polynomials $\check{\boldsymbol{a}} = [\mathbf{a}_1, \dots, \mathbf{a}_m] \in R^m$, we denote by $h_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}})$ the random variable $\check{\boldsymbol{a}} \cdot \check{\boldsymbol{e}} = \sum_{i \in [l]} \mathbf{a}_i \otimes \mathbf{e}_i \in R$, where $\check{\boldsymbol{e}} = (\mathbf{e}_1, \dots, \mathbf{e}_m) \leftarrow \mathcal{D}^{mn}$. If $\check{\boldsymbol{a}}$ is chosen from R^m uniformly at random, then the statistical distance to uniformity of $(\check{\boldsymbol{a}}, H_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}}))$ is at most*

$$\frac{1}{2}\sqrt{\prod_{i\in[t]}\left(1+\left(\frac{|\mathbb{F}|}{|\mathcal{D}|^l}\right)^{\deg(\mathbf{f}_i)}\right)-1},$$

where $\mathbf{f} = \prod_{i \in [t]} \mathbf{f}_i$ is the factorization of $\mathbf{f}$ over $\mathbb{F}$.

To show this, we need the following lemma.

Lemma 4.4.4 (Lemma 4.4 in [Mic07])**.** *Let R be a finite ring, and $\boldsymbol{z} = (z_1, \dots, z_m) \in R^m$ a vector of arbitrary ring elements. If $\boldsymbol{a} = [a_1, \dots, a_m] \leftarrow R^m$, then $\boldsymbol{a} \cdot \boldsymbol{z} = \sum_{i \in [m]} a_i \cdot z_i \in R$ is uniformly distributed over the ideal $\langle z_1, \dots, z_m \rangle$. In particular, for any $\boldsymbol{z}$,*

$$\Pr_{\boldsymbol{a} \leftarrow R^m}\left[\sum_{i\in[m]} a_i \cdot z_i = 0\right] = \frac{1}{|\langle z_1, \dots, z_m \rangle|}.$$

To contain itself, we give the proof.

Proof. We start the proof following Micciancio's proof [Mic07]. For simplicity, we let q be the cardinality of $\mathbb{F}$. In order to show the theorem, we bound the collision probability of $(\check{\boldsymbol{a}}, H_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}}))$.

$$\begin{aligned}\Pr_{\check{\boldsymbol{a}},\check{\boldsymbol{a}}',\check{\boldsymbol{e}},\check{\boldsymbol{e}}'}[\check{\boldsymbol{a}} = \check{\boldsymbol{a}}' \wedge H_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}}) = H_{\check{\boldsymbol{a}}'}(\check{\boldsymbol{e}}')] &= \Pr_{\check{\boldsymbol{a}},\check{\boldsymbol{a}}}[\check{\boldsymbol{a}} = \check{\boldsymbol{a}}'] \cdot \Pr_{\check{\boldsymbol{a}},\check{\boldsymbol{a}}',\check{\boldsymbol{e}},\check{\boldsymbol{e}}'}[H_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}}) = H_{\check{\boldsymbol{a}}'}(\check{\boldsymbol{e}}') \mid \check{\boldsymbol{a}} = \check{\boldsymbol{a}}'] \\ &= \frac{1}{q^{mn}} \cdot \Pr_{\check{\boldsymbol{a}},\check{\boldsymbol{e}},\check{\boldsymbol{e}}'}\left[\sum_{i\in[m]} \mathbf{a}_i \otimes (\mathbf{e}_i - \mathbf{e}'_i) = 0\right].\end{aligned}$$

Fix $\check{\boldsymbol{e}}$ and $\check{\boldsymbol{e}}'$. Then, by the above lemma (Lemma 4.4.4), the probability that $\sum_{i\in[m]} \mathbf{a}_i \otimes (\mathbf{e}_i - \mathbf{e}'_i) = 0$ equals to $1/\left|\langle \mathbf{e}_1 - \mathbf{e}'_1, \dots, \mathbf{e}_m - \mathbf{e}'_m \rangle\right|$. Let $\mathcal{I}$ be the set of all ideals of $R = \mathbb{F}[x]/\langle \mathbf{f} \rangle$. Hence, conditioning on the ideals, we have that

$$\begin{aligned}
\frac{1}{q^{mn}} \cdot \Pr_{\breve{a},\breve{e},\breve{e}'}\left[\sum_{i\in[m]} \mathbf{a}_i \otimes (\mathbf{e}_i - \mathbf{e}'_i) = 0\right] &= \frac{1}{q^{mn}} \cdot \sum_{I\in\mathcal{I}} \frac{1}{|I|} \cdot \Pr_{\breve{e},\breve{e}'}\left[\langle \mathbf{e}_1 - \mathbf{e}'_1, \dots, \mathbf{e}_m - \mathbf{e}'_m \rangle = I\right] \\
&\le \frac{1}{q^{mn}} \cdot \sum_{I\in\mathcal{I}} \frac{1}{|I|} \cdot \Pr_{\breve{e},\breve{e}'}\left[\langle \mathbf{e}_1 - \mathbf{e}'_1, \dots, \mathbf{e}_m - \mathbf{e}'_m \rangle \subseteq I\right] \\
&= \frac{1}{q^{mn}\cdot q^n} \cdot \sum_{I\in\mathcal{I}} \frac{q^n}{|I|} \cdot \prod_{i\in[m]} \Pr_{\mathbf{e}_i,\mathbf{e}'_i}[\mathbf{e}_i - \mathbf{e}'_i \in I].
\end{aligned}$$

Since $\mathbb{F}$ is a field, $\mathbb{F}[x]$ is a principal ideal domain. Then, any ideal $I \in \mathcal{I}$ of $R = \mathbb{F}[x]/\langle \mathbf{f} \rangle \simeq \mathbb{F}[x]/\langle \mathbf{f}_1 \rangle \times \cdots \times \mathbb{F}[x]/\langle \mathbf{f}_t \rangle$ are of the form $\langle \mathbf{p} \rangle$ where $\mathbf{p}$ is a factor of $\mathbf{f}$. For any subset $S \subseteq [t]$, let $\mathbf{p}_S = \prod_{i\in S} \mathbf{f}_i$. The ideals of R are $\mathcal{I} = \{\langle \mathbf{p}_S \rangle \mid S \subseteq [t]\}$. (Micciancio restricted the argument in the case where $\mathbf{f} = x^n - 1$, but this argument can be applied to any monic polynomial $\mathbf{f}$ as in the above.) In addition, note that the ideal $\langle \mathbf{p}_S \rangle \simeq \prod_{i\in[t]\setminus S} \mathbb{F}[x]/\langle \mathbf{f}_i \rangle$. Hence, we have $|\langle \mathbf{p}_S \rangle| = q^{n-\deg(\mathbf{p}_S)}$. Therefore, we have that

$$\begin{aligned}
\Pr_{\mathbf{e}_i,\mathbf{e}'_i}[\mathbf{e}_i - \mathbf{e}'_i \in \langle \mathbf{p}_S \rangle] &= \Pr_{\mathbf{e}_i,\mathbf{e}'_i}[\mathbf{e}_i \equiv \mathbf{e}'_i \pmod{\mathbf{p}_S}] \\
&\le \max_{\mathbf{e}} \Pr_{\mathbf{e}_i}[\mathbf{e}_i \equiv \mathbf{e} \pmod{\mathbf{p}_S}] && (4.1) \\
&\le \frac{1}{|\mathcal{D}|^{\deg(\mathbf{p}_S)}}, && (4.2)
\end{aligned}$$

where $\mathbf{e}$ ranges over $\mathbb{F}^{\deg(\mathbf{p}_S - 1)}$.

Using this bound, we obtain

$$\frac{q^n}{|\langle \mathbf{p}_S \rangle|} \prod_{i\in[m]} \Pr_{\mathbf{e}_i,\mathbf{e}'_i}[\mathbf{e}_i - \mathbf{e}'_i \in \langle \mathbf{p}_S \rangle] \le \frac{q^n}{q^{n-\deg(\mathbf{p}_S)}} \left(\frac{1}{|\mathcal{D}|^{\deg(\mathbf{p}_S)}}\right)^m = \left(\frac{q}{|\mathcal{D}|^m}\right)^{\deg(\mathbf{p}_S)}.$$

By summing up, we have that

$$\sum_{\langle \mathbf{p}_S \rangle \in \mathcal{I}} \frac{q^n}{|\langle \mathbf{p}_S \rangle|} \prod_{i\in[m]} \Pr_{\mathbf{e}_i,\mathbf{e}'_i}[\mathbf{e}_i - \mathbf{e}'_i \in \langle \mathbf{p}_S \rangle] \le \sum_{S\subseteq[t]} \left(\frac{q}{|\mathcal{D}|^m}\right)^{\deg(\mathbf{p}_S)} = \prod_{i\in[t]} \left(1 + \left(\frac{q}{|\mathcal{D}|^m}\right)^{\deg(\mathbf{f}_i)}\right).$$

Combining them, we obtain that

$$\Pr_{\breve{a},\breve{a}',\breve{e},\breve{e}'}\left[(\breve{a}, H_{\breve{a}}(\breve{e})) = (\breve{a}', H_{\breve{a}'}(\breve{e}'))\right] \le \frac{1}{q^{mn+n}} \prod_{i\in[t]} \left(1 + \left(\frac{q}{|\mathcal{D}|^m}\right)^{\deg(\mathbf{f}_i)}\right).$$

Applying the bound lemma, we conclude that

$$\Delta((\breve{a}, H_{\breve{a}}(\breve{e})), (\breve{a}', \mathbf{u})) \le \frac{1}{2}\sqrt{\prod_{i\in[t]} \left(1 + \left(\frac{q}{|\mathcal{D}|^m}\right)^{\deg(\mathbf{f}_i)}\right) - 1},$$

where $\breve{a}' \leftarrow R^m$ and $\mathbf{u} \leftarrow R$. □

We let apply the regularity lemma to several cases. Let

$$\Delta(q, \mathbf{f}, d) = \sqrt{\prod_{i \in [t]} \left(1 + \left(\frac{q}{d^m}\right)^{\deg(\mathbf{f}_i)}\right) - 1}.$$

1. If $\mathbf{f}$ is irreducible over $\mathbb{F}$, we have that $\Delta(q, \mathbf{f}, d) = \sqrt{(q/d^m)^n} = 2^{\frac{1}{2}(\log q - mn \log d)}$. By setting, $m > (1+\delta) \log q / \log d$ for $\delta > 0$, we obtain the upper bound $2^{-\frac{1}{2}\delta n \log q} = 2^{-\Omega(n)}$. It indicates if we set $m = O(1)$ and $q = \mathrm{poly}(n)$ satisfying the above then we have $\Delta(q, \mathbf{f}, d)$ is negligible even if $d = 2$ or 3.
2. If $\mathbf{f} = \mathbf{f}_1 \cdot \mathbf{f}_2$, where $\deg(\mathbf{f}_i) = n/2$, we have that $\Delta(q, \mathbf{f}, d) = \sqrt{(1 + (q/d^m)^{n/2})^2 - 1} = \sqrt{2(q/d^m)^{n/2} + (q/d^m)^n}$. By setting, $m > (1 + \delta) \log q / \log d$ for $\delta > 0$, we obtain the upper bound
$$\sqrt{3} \cdot (q/d^m)^{n/4} \leq 2^{-\frac{1}{4}\delta n \log q + \frac{1}{2} \log 3} = 2^{-\Omega(n)}.$$
3. If $\mathbf{f}$ is completely split over $\mathbb{F}$, we have that $\Delta(q, \mathbf{f}, d) = \sqrt{(1 + q/d^m)^n - 1}$. Suppose that $m = (1+\delta) \log q / \log d$ for some $\delta > 0$ to set $q/d^m < 1$ sufficiently small. Suppose that $q = n^a$ for some $a > 0$. Then, we have the upper bound $\sqrt{2nq/d^m} = 2^{\frac{1}{2}(1+(a+1)\log n - m \log d)}$, since $(1 + q/d^m)^n \leq 1 + 2nq/d^m$. To set the upper bound negligible in n, we need to have $m \log d = \omega(\log n)$, e.g., $m = \omega(\log n)$ and $d = O(1)$ or $m = O(1)$ and $d = \omega(\log n)$, which sacrifices the efficiency of ILHash.

In particular, we have the following lemma for $\mathbf{f} = x^n + 1$ and $q \equiv 3 \pmod 8$ by Lemma 3.1.2 and the discussion (2).

Lemma 4.4.5 ([SSTX09])**.** *Let $\mathbf{f} = x^n + 1$ and $n = 2^k$, where $k \geq 2$. Let $m > (1 + \delta) \log q / \log d$ for some constant $\delta > 0$. If q is a prime with $q \equiv 3 \pmod 8$, the statistical distance $\frac{1}{2}\Delta(q, x^n + 1, d)$ of $(\check{\mathbf{a}}, h_{\check{\mathbf{a}}}(\check{\mathbf{e}}))$ from the uniform is at most $2^{-\frac{1}{4}\delta n \log q}$.*

The analysis of the last case (3) is improved if we improve the inequality (4.1), which is not tight and overkills to obtain a good bound. Some experiments by the author indicates there is more tight bound for the case where $\mathbf{f}$ splits $\mathbb{F}$ completely. However, we fail to prove the good bound. In addition, the above regularity lemma states only the case $\mathbb{F}$ or $\mathbb{Z}_q$ for a prime q. Another possible extension is for $\mathbb{Z}_q$, where q is a composite. We leave obtaining these bound results as an open problem.

5 Commitment

In this chapter, we construct simple string commitment schemes based on lattice problems: They are statistically hiding and computationally binding (see the definition below). We only consider string commitment schemes in the trusted setup model.

Organization: Section 5.1 defines model and the security notions on non-interactive commitment schemes. In Section 5.2, we review the Halevi–Micali commitment scheme as the general construction from a collision-resistant hash scheme. Section 5.3 and Section 5.4 reviews the lattice-based commitment scheme which is proposed by Kawachi et al. [KTX08].

In addition, Fujisaki [Eii08] pointed out that our commitment scheme can be converted into a chameleon hash scheme or a trapdoor commitment scheme, by adjusting the parameters and replacing some functions. This construction also appears in Peikert [Pei09b, Section 2.2]. We will argue this as trapdoor hash in Section 10.10.

5.1 Definitions

We consider a non-interactive string commitment scheme in the trusted setup model. The trusted setup model is often required to construct practically efficient cryptographic schemes such as non-interactive string commitment schemes. In this model, we assume that a trusted party $\mathcal{T}$ honestly sets up a system parameter for the sender and the receiver.

Let us specify how it works. First $\mathcal{T}$ generates public parameters and distribute them to users. Both parties, the sender and the receiver, then share public parameters. The scheme runs in two phase, called committing and revealing phases. In the committing phase, the sender commits his/her message, say a string s. He/she generates a commitment string *cmt* and an open value *ov*, and sends *cmt* to the receiver. In the revealing phase, the sender gives the receiver the decision s and the open string *ov*. The receiver verifies the validity of *cmt* with *msg* and *ov*.

We require two security notions of the string commitment schemes, statistically-hiding and computationally-binding properties. Intuitively, we say that the commitment scheme is statistically hiding, if any computationally unbounded adversarial receiver cannot distinguish two commitment strings generated from two distinct strings. Also, it is computationally binding, if any polynomial-time adversarial sender cannot change the committed string after sending the commitment.

5.1.1 Model of Non-Interactive Commitment Schemes

Let $\mathsf{NIC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Ver})$ over a message space M_n be a non-interactive commitment scheme. Notation of the algorithms are below:

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs public parameters *param*.

$\mathsf{Com}(param, msg)$**:** A commitment algorithm, given *param* and a value $msg \in M_n$, outputs a commitment *cmt* and a value *ov*.

$\mathsf{Ver}(param, cmt, msg, ov)$**:** A verification algorithm, given *param*, *cmt*, *msg*, and *ov*, returns 0 (reject) or 1 (accept).

Often, we say NIC is a bit commitment scheme if $M_n = \{0, 1\}$. We say NIC is a string commitment scheme if $M_n = \{0, 1\}^{l(n)}$ for $l(n) \neq \omega(1)$.

The correctness of the commitment is defined as follows: For any $msg \in M_n$, *param* generated by $\mathsf{Setup}(1^n)$ and (cmt, ov) generated by a valid committee , the verifier always accepts $param, cmt, msg, ov$. Formally, it holds that for any $msg \in M_n$,

$$\Pr\left[b' = 1 : \begin{array}{c} param \leftarrow \mathsf{Setup}(1^n); \\ (cmt, ov) \leftarrow \mathsf{Com}(param, msg); \\ b \leftarrow \mathsf{Ver}(param, cmt, msg, ov); \end{array} \right] = 1,$$

where the probability is taken over coins of Setup and Com.

5.1.2 Security Notions

To define the security notion, consider the experiments $\mathbf{Exp}^{\mathrm{bind}}_{\mathsf{NIC},\mathcal{A}}(n)$ and $\mathbf{Exp}^{\mathrm{hide}}_{\mathsf{NIC},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\mathrm{bind}}_{\mathsf{NIC},\mathcal{A}}(n)$**:**

Setup Phase: The challenger C runs $\mathsf{Setup}(1^n)$ and obtains *param*. The adversary $\mathcal{A}$ is given the security parameter 1^n and the parameters *param*.

Challenge Phase: The adversary outputs cmt, (msg, ov), and (msg', ov'). If $msg, msg' \in M_n$, $msg \neq msg'$, $\mathsf{Ver}(param, cmt, msg, ov) = 1$, and $\mathsf{Ver}(param, cmt, msg', ov') = 1$ then C returns 1. Otherwise, it returns 0.

Experiment $\mathbf{Exp}^{\mathrm{hide}}_{\mathsf{NIC},\mathcal{A}}(n)$:

Setup Phase: The challenger C runs $\mathsf{Setup}(1^n)$ and obtains $param$. The adversary $\mathcal{A}$ is given the security parameter 1^n and the parameters $param$.

Challenge Phase: The adversary outputs msg_0 and msg_1. If $msg_0, msg_1 \in M_n$ and $msg_0 \neq msg_1$, the challenger flips a fair coin $b \leftarrow \{0, 1\}$, generates $cmt^* \leftarrow \mathsf{Com}(param, msg_b)$, and sends cmt^* to the adversary. Otherwise, C returns 0 and halts.

Decision Phase: Finally, the adversary outputs its decision b'. If $b = b'$ the challenger returns 1, otherwise 0.

Here, the security notions of the non-interactive commitment schemes we require can be formalized as follows:

Definition 5.1.1 (Hiding property)**.** Consider a non-interactive commitment scheme $\mathsf{NIC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Ver})$.

We say NIC is perfectly hiding if any two messages $msg, msg' \in M_n$, $(param, cmt_{msg})$ and $(param, cmt_{msg'})$ are equally distributed, where $param \leftarrow \mathsf{Setup}(1^k)$, $(cmt_{msg}, ov_{msg}) \leftarrow \mathsf{Com}(param, msg)$, and $(cmt_{msg'}, ov_{msg'}) \leftarrow \mathsf{Com}(param, msg')$.

We say NIC is statistically hiding if any two messages $msg, msg' \in M_n$, the statistical distance between $(param, cmt_{msg})$ and $(param, cmt_{msg'})$ is negligible in n.

Let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{NIC},\mathcal{A}}(n) := \left| \Pr\left[\mathbf{Exp}^{\mathrm{hide}}_{\mathsf{NIC},\mathcal{A}}(n) = 1\right] - \frac{1}{2} \right|.$$

We say NIC is computationally hiding if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{NIC},\mathcal{A}}(n)$ is negligible in n.

Definition 5.1.2 (Binding property)**.** Let $\mathsf{NIC} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Ver})$ be a non-interactive commitment scheme. Let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathrm{bind}}_{\mathsf{NIC},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{bind}}_{\mathsf{NIC},\mathcal{A}}(n) = 1\right].$$

We say a non-interactive commitment scheme NIC is computationally binding if $\mathbf{Adv}^{\mathrm{bind}}_{\mathsf{NIC},\mathcal{A}}(n)$ is negligible in n for any polynomial-time adversary $\mathcal{A}$.

5.1.3 Special Property

In addition, we define a special property of non-interactive commitment. We say a non-interactive commitment NIC is special if the scheme can be modeled as follows:

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs public parameters *param*. The parameter *param* defined the function $Com_{param} : M_n \times D_n \to R_n$.

$\mathsf{Com}(param, msg)$**:** A commitment algorithm, given *param* and a value $msg \in M_n$, first generate a randomness $r \in D_n$, and outputs a commitment $cmt = Com_{param}(msg, r)$ and a value $ov = (msg, r)$.

$\mathsf{Ver}(param, cmt, msg, ov)$**:** A verification algorithm, given *param*, *cmt*, *msg*, and *ov*, returns 1 if $cmt = Com_{param}(msg, ov)$ and 0 otherwise.

5.2 Example: The Halevi–Micali Commitment Scheme

General constructions of statistically-hiding and computationally-binding string commitment schemes are known from a family of collision-resistant hash functions [DPP97, HM96]. Their constructions used universal hash functions for the statistically-hiding property and very similar to each others. For simplicity, we decide to review the Halevi-Micali construction.

The Halevi–Micali commitment scheme: Halevi and Micali proposed a simple string commitment scheme based on the collision-resistance hash function [HM96], which is very similar to one in [DPP98].

Let n denote the security parameter and let l be a positive integer at least $6n+4$. Let $\mathcal{H}_n = \{h_k : \{0,1\}^* \to \{0,1\}^n\}_{k \in K_n}$ be a family of collision-free hash functions and $\mathcal{H} = \{\mathcal{H}_n\}$ a hash family. Let $\mathsf{Hash} = (\mathsf{Setup}, \mathsf{Eval})$ be a corresponding hash scheme. Let $\mathcal{F}_n = \{f : \{0,1\}^l \to \{0,1\}^n\}$ be a family of universal hash functions. For example, we can use $\{f_{a_1,\ldots,a_6} : \mathrm{GF}(2^n)^7 \to \mathrm{GF}(2^n) : a_i \in \mathrm{GF}(2^n)\}$, where $f_{a_1,\ldots,a_6}(s_0, \ldots, s_6) = s_0 + a_1 s_1 + \cdots + a_6 s_6$ and each n-bit string s_i is interpreted as an element in $\mathrm{GF}(2^n)$.

The Halevi–Micali commitment scheme $(\mathsf{Setup}', \mathsf{Com}', \mathsf{Ver}')$ is defined as follows:

Scheme 5.2.1. For simplicity, we set $l = 7n$.

$\mathsf{Setup}'(1^n)$**:** The setup algorithm obtains $a \leftarrow \mathsf{Setup}(1^n)$ and outputs $param = a$.

$\mathsf{Com}'(a, msg)$**:** The commitment algorithm computes $s \leftarrow h_a(msg)$, picks a random $r \in \{0,1\}^l$, computes $y \leftarrow h_a(r)$, and picks a random function $f \in \mathcal{F}$ for which $f(r) = s$. Then it outputs $(cmt, ov) = ((f, y), r)$.

$\mathsf{Ver}'(a, (f, y), msg, r)$**:** The verification algorithm accepts if $y = h_a(r)$ and $f(r) = h_a(msg)$. Otherwise, rejects.

To pick a random function $f_{a_1,\ldots,a_6}$ such that $r_0 + a_1 r_1 + \cdots + a_6 r_6 = s$, it computes as follows: Choose random elements $a_1, \ldots, a_5$ and compute $a_6 = r_6^{-1}(s - (r_0 + a_1 r_1 + \cdots + a_5 r_5))$ if $r_6 \neq 0$. If $r_6 = 0$, choose a_6 at random.

Halevi and Micali showed that the scheme is computationally binding if H is collision resistant and statistically hiding with the distance 2^{-n}. Note that the length of commitment is $|y| + |f| = n + 6n = 8n$ and the length of decommitment is $|m| + |r| = |m| + 7n$ if we use the above universal hash functions. We will employ this commitment scheme in Chapter 9.

5.3 A Lattice-based String Commitment Scheme

Here, we review a more direct and simpler construction from the lattice-based hash functions without the universal hash functions [KTX08]. The input of the commitment function is an m-bit vector $\boldsymbol{x}$ obtained by concatenating a random string $\rho = (\rho_1, \dots, \rho_{m/2}) \in \{0,1\}^{m/2}$ and a message string $s = (s_1, \dots, s_{m/2}) \in \{0,1\}^{m/2}$, i.e., $\boldsymbol{x} = \rho \circ s \in \{0,1\}^m$. We then define the commitment function on inputs s and ρ as

$$Com_{\boldsymbol{A}}(s;\rho) := h_{\boldsymbol{A}}(\boldsymbol{x}).$$

We define our non-interactive string commitment scheme LNIC by using the above commitment function:

Scheme 5.3.1 (LNIC, [KTX08])**.**

$\mathsf{Setup}(1^n)$**:** Given input 1^n, the algorithm samples $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and outputs $param = \boldsymbol{A}$.

$\mathsf{Com}(\boldsymbol{A}, msg = s; \rho)$**:** Given inputs $\boldsymbol{A}$ and $s \in \{0,1\}^{m/2}$, the algorithm samples $\rho \leftarrow \{0,1\}^{m/2}$. It computes $\boldsymbol{c} \leftarrow Com_{\boldsymbol{A}}(s;\rho)$, and outputs $cmt = \boldsymbol{c}$ and $ov = \rho$.

$\mathsf{Ver}(\boldsymbol{A}, \boldsymbol{c}, s, \rho)$**:** The algorithm checks that $s, \rho \in \{0,1\}^{m/2}$ and $Com_{\boldsymbol{A}}(s;\rho) = \boldsymbol{c}$. It outputs 1 if the checks are passed, 0 otherwise.

Lemma 5.3.2. *If q is a prime and $m > 2n(1+\delta)\log q$ for some constant δ, if $\mathrm{SIS}_{q,m,\sqrt{m}}$ is hard on the average, then $Com_{\boldsymbol{A}}$ is a statistically-hiding and computationally-binding string commitment scheme in the trusted set up model.*

Proof. The computationally-binding property immediately follows from the collision-resistant property. We now show the statistically-hiding property.

Let $\boldsymbol{A} = [\boldsymbol{a}_1 \cdots \boldsymbol{a}_m]$. We then have $Com_{\boldsymbol{A}}(s;\rho) = \sum_{i=1}^{m/2} \rho_i \boldsymbol{a}_i + \sum_{i=1}^{m/2} s_i \boldsymbol{a}_{i+m/2}$. Applying the leftover hash lemma, we can say that a random subset sum of $\boldsymbol{a}_i$ is statistically close to the uniform distribution for almost all choices of $\boldsymbol{a}_i$.

In our proof, we consider $\mathbb{Z}_q^n$ as a finite Abelian group G. Since $m > 2n(1 + \delta)\log q$, we have that

$$\left(\frac{|G|}{2^{m/2}}\right)^{1/4} \leq q^{-\delta n/4}.$$

Thus, by Lemma 4.3.3, for all but an at most $q^{-\delta n/4}$ fraction of $\boldsymbol{A} = [\boldsymbol{a}_1, \dots, \boldsymbol{a}_m] \in \mathbb{Z}_q^{n \times m}$, we have that $\Delta(\boldsymbol{u}, \sum_{i \in [m/2]} \rho_i \boldsymbol{a}_i) \leq q^{-\delta n/4}$, where $\boldsymbol{u} \in \mathbb{Z}_q^n$ is uniform random variable. Assume that we have such $\boldsymbol{A}$. So, we have $\Delta(\boldsymbol{u}, Com_{\boldsymbol{A}}(0^{m/2};\rho)) \leq q^{-\delta n}$.

By the definition of $Com_{\boldsymbol{A}}$, for any $s \in \{0,1\}^{m/2}$, we have $\Delta(\boldsymbol{u}, Com_{\boldsymbol{A}}(s;\rho)) \leq q^{-\delta n/4}$. By the triangle inequality, we obtain

$$\begin{aligned}\Delta(Com_{\boldsymbol{A}}(s_1;\rho_1), Com_{\boldsymbol{A}}(s_2;\rho_2)) &\leq \Delta(\boldsymbol{u}, Com_{\boldsymbol{A}}(s_1;\rho_2)) + \Delta(\boldsymbol{u}, Com_{\boldsymbol{A}}(s_2;\rho_2)) \\ &\leq 2q^{-\delta n/4},\end{aligned}$$

for any message s_1 and s_2. This shows that, for all but negligible fraction of choice of $\boldsymbol{A}$, distributions of two commitments are statistically close.

□

5.3.1 Extending the Domain

Notice that the message space of commitment function is simply extended by the using combining collision-resistant hash function $h : \{0,1\}^* \to \{0,1\}^{m/2}$ [KR00]. New commitment function Com' is defined by

$$Com'(s;\rho) = Com(h(s);\rho).$$

We here take more direct way to extend the domain. The spirit of the proof is essentially same as that of Krawczyk and Rabin [KR00]. Using the Merkle–Damgård technique [Mer89, Dam89], we obtain a string commitment scheme whose commitment function is $Com_{\boldsymbol{A}} : \{0,1\}^* \times \{0,1\}^{m/2} \to \mathbb{Z}_q^n$ rather than $Com_{\boldsymbol{A}} : \{0,1\}^{m/2} \times \{0,1\}^{m/2} \to \mathbb{Z}_q^n$ as the following.

Assume that $m = 2r$. Let $\boldsymbol{A} = [\boldsymbol{B}|\boldsymbol{C}]$, where $\boldsymbol{B}, \boldsymbol{C} \in \mathbb{Z}_q^{n\times r}$. For $\boldsymbol{X} \in \mathbb{Z}_q^{n\times r}$, we define $f_{\boldsymbol{X}} : \{0,1\}^r \to \mathbb{Z}_q^n$ as the hash function $f_{\boldsymbol{X}}(\boldsymbol{s}) = \boldsymbol{X}\boldsymbol{s} \bmod q$.

To apply the Merkle–Damgård technique, we need two utility functions in our case. Let l be $\lceil n \log q \rceil$ and let $t : \mathbb{Z}_q^n \to \{0,1\}^l$ be some one-to-one function that we compute t and t^{-1} efficiently, say $t(\boldsymbol{a})$ outputs a binary string representing $\sum_{i\in[n]} a_i q^{i-1}$. Next, let $\mathsf{pad} : \{0,1\}^* \to \{0,1\}^*$ be the padding function for the Merkle-Damgård construction. We employ Merkle's padding function pad pads with 0 and adds the length information to the original message. Let $0 < b < r - l$ be the length parameter. Formally, if we have a block compression function $f : \{0,1\}^r \to \{0,1\}^l$ the padding function works on input $s \in \{0,1\}^*$ of length $0 < a < 2^b$ as follows.

$$\mathsf{pad}(s) = s \circ 10^d \circ \mathrm{len}_b(a),$$

where d is the smallest non-negative integer such that $a + d + b + 1$ is a multiple of $r - l$, b is some fixed integer $\mathrm{len}_b(a)$ denotes the b-bit representation of the integer a.

Applying the Merkle–Damgård construction to $f_{\boldsymbol{C}}$, we obtain new hash function $h_{\boldsymbol{C}} : \{0,1\}^* \to \mathbb{Z}_q^n$. The precise definition of $h_{\boldsymbol{C}}$ is as follows:

1. On input s, obtain a padded message $S \leftarrow \mathsf{pad}(s)$
2. Chop it into $(S_0, \ldots, S_k)$, where $S_i \in \{0,1\}^{r-l}$
3. Let $H_0 = \boldsymbol{0}$ (more generally, some fixed IV can be used)

4. For $i = 1$ to $k + 1$ do $H_i \leftarrow f_{\boldsymbol{C}}(t(H_{i-1}) \circ S_{i-1})$
5. Output H_{k+1}

Our new commitment scheme is defined as follows: for $s \in \{0,1\}^*$ and $\rho \in \{0,1\}^r$,

$$Com_{\boldsymbol{A}}(s;\rho) := h_{\boldsymbol{C}}(s) + f_{\boldsymbol{B}}(\rho) \bmod q.$$

Lemma 5.3.3. *If there exists a polynomial-time machine outputting a collision for $Com_{\boldsymbol{A}}$, then there exists a polynomial-time machine outputting a collision for $f_{\boldsymbol{A}}$.*

Proof. Let us assume that we obtain a collision $(s,\rho),(\tilde{s},\tilde{\rho}) \in \{0,1\}^* \times \{0,1\}^r$ for $Com_{\boldsymbol{A}}$. By the assumption, we have

$$h_{\boldsymbol{C}}(s) + f_{\boldsymbol{B}}(\rho) \equiv h_{\boldsymbol{C}}(\tilde{s}) + f_{\boldsymbol{B}}(\tilde{\rho}) \pmod{q}.$$

If $\rho = \tilde{\rho}$, we have $s \neq \tilde{s}$ and $h_{\boldsymbol{C}}(s) = h_{\boldsymbol{C}}(\tilde{s})$. Using the reduction for the Merkle-Damgård construction (see e.g., [KL07, Thm. 4.14]), we obtain $u \neq \tilde{u} \in \{0,1\}^r$ such that $f_{\boldsymbol{C}}(u) = f_{\boldsymbol{C}}(\tilde{u})$. Thus, we have a collision $u \circ \rho, \tilde{u} \circ \rho \in \{0,1\}^{2r}$ for $f_{\boldsymbol{A}}$.

Next, we assume that $\rho \neq \tilde{\rho}$. Let S and $\tilde{S}$ be padded messages of s and $\tilde{s}$, respectively. Assume that S and $\tilde{S}$ are chopped into $(S_0, \ldots, S_k)$ and $(\tilde{S}_0, \ldots, \tilde{S}_{k'})$, respectively. Let H_k and $\tilde{H}_{k'}$ be inner hash values for s and $\tilde{s}$ in the algorithm, respectively. By the definition of H_k and $\tilde{H}_{k'}$, we obtain

$$h_{\boldsymbol{C}}(s) = f_{\boldsymbol{C}}(t(H_k) \circ S_k),$$
$$h_{\boldsymbol{C}}(\tilde{s}) = f_{\boldsymbol{C}}(t(\tilde{H}_{k'}) \circ \tilde{S}_{k'}).$$

Combining the above equations with the assumption, we obtain

$$f_{\boldsymbol{A}}(t(H_k) \circ S_k \circ \rho) = f_{\boldsymbol{A}}(t(\tilde{H}_{k'}) \circ \tilde{S}_{k'} \circ \tilde{\rho}).$$

So, we have a collision $t(H_k) \circ S_k \circ \rho$ and $t(\tilde{H}_{k'}) \circ \tilde{S}_{k'} \circ \tilde{\rho} \in \{0,1\}^{2r}$ for $f_{\boldsymbol{A}}$. □

We use this commitment scheme in the rest of the paper. We often abuse the notation of $Com_{\boldsymbol{A}}$. For example $Com_{\boldsymbol{A}}(v_1, v_2;\rho)$ denotes $Com_{\boldsymbol{A}}(\text{string}(v_1) \circ \text{string}(v_2);\rho)$, where string$(v)$ is a binary representation of v.

5.4 An Ideal-Lattice-Based String Commitment Scheme

Using ILHash in Section 4.4 we also obtain a simple string commitment scheme. We first extend the notation of *Com*: For $\check{\boldsymbol{a}} \in R_{\mathbf{f},q}^m$,

$$Com_{\check{\boldsymbol{a}}}(\cdot) = Com_{\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})}(\cdot).$$

We define our non-interactive string commitment scheme ILNIC.

Scheme 5.4.1 (ILNIC, [KTX08]).

$\mathsf{Setup}(1^n)$**:** Given input 1^n, the algorithm samples $\check{\boldsymbol{a}} \in R^m_{\mathbf{f},q}$ and outputs $param = \check{\boldsymbol{a}}$.

$\mathsf{Com}(\check{\boldsymbol{a}}, msg = s; \rho)$**:** Given inputs $\boldsymbol{A}$ and $s \in \{0,1\}^{mn/2}$, the algorithm samples $\rho \leftarrow \{-1, 0, +1\}^{mn/2}$. It computes $\mathbf{c} \leftarrow Com_{\check{\boldsymbol{a}}}(s; \rho)$, and outputs $cmt = \mathbf{c}$ and $ov = \rho$.

$\mathsf{Ver}(\check{\boldsymbol{a}}, \mathbf{c}, s, \rho)$**:** The algorithm checks that $s \in \{0,1\}^{mn/2}$, $\rho \in \{-1, 0, +1\}^{mn/2}$ and $Com_{\check{\boldsymbol{a}}}(s; \rho) = \mathbf{c}$. It outputs 1 if the checks are passed, 0 otherwise.

We apply Micciancio's regularity lemma to ILHash and obtain the statistically-hiding property of a string commitment scheme. Straightforwardly, the computational-binding property follows from the collision-resistant property of the underlying hash function. Formally, we obtain the following lemma as in Lemma 5.3.2.

Lemma 5.4.2. *Let $\mathbf{f} \in \mathbb{Z}[x]$ be a monic and irreducible polynomial of degree n. Let q be a prime polynomially bounded by n. Let $\mathbf{f} = \prod_{i \in [t]} \mathbf{f}_i$ is the factorization of $\mathbf{f}$ over $\mathbb{Z}_q$. Let*

$$\Delta = \frac{1}{2}\Delta(q, \mathbf{f}, 3) = \frac{1}{2}\sqrt{\prod_{i \in [t]}\left(1 + \left(\frac{q}{3^m}\right)^{\deg(\mathbf{f}_i)}\right) - 1},$$

defined in Section 4.4.2. The scheme ILNIC *is a statistically-hiding and computationally-binding string commitment scheme in the trusted setup model if $\mathbf{f}\text{-SIS}^{\infty}_{q,m,1}$ is hard on average and if Δ is negligible in n.*

Furthermore, let $E_3 = \mathrm{EF}_\infty(\mathbf{f}, 3)$. Let $m > 4\log q$ and $q > 3E_3 mn^{3/2}\log n$. Then, for $\gamma = 8E_3^2 mn \log^2 n$, if $\mathbf{f}\text{-SVP}^{\infty}_{\gamma}$ is hard in the worst case and Δ is negligible in n, the scheme ILNIC *is a statistically-hiding and computationally-binding string commitment scheme.*

In particular, let $\mathbf{f} = x^{2^k} + 1$ with $k \geq 2$ and $q \equiv 3 \bmod 8$, the scheme is statistically hiding by Lemma 4.4.5.

Using the Merkle-Damgård technique, we obtain the string commitment scheme whose commitment function is $Com_{\boldsymbol{A}} : \{0,1\}^* \times \{0,1\}^{mn/2} \to \mathbb{Z}_q^n$ rather than $Com_{\boldsymbol{A}} : \{0,1\}^{mn/2} \times \{0,1\}^{mn/2} \to \mathbb{Z}_q^n$.

6 Identification

This chapter contains identification (ID) schemes based on lattice problems and new security proofs for the variants of the Micciancio-Vadhan ID scheme.

Organization: Section 6.1 introduces public-key identification, the construction idea, and comparisons. Section 6.2 reviews the definitions of identification schemes. We review the several identification schemes in this chapter. In Section 6.3, we review the Micciancio and Vadhan protocol. The ID schemes based on them are in Section 6.4. Section 6.5 reviews the identification scheme given by Lyubashevsky. Section 6.6 reviews Stern's protocol and, based on it, we review the Kawachi–Tanaka–Xagawa identification scheme Section 6.7. Finally, we review the new Lyubashevsky identification in Section 6.8.

6.1 Introduction

We have already noted hash schemes and commitment schemes in the previous chapters (Chapter 4 and Chapter 5). We next describe the identification (ID) schemes based on lattice problems, which are directly based on the lattice-based hash schemes.

Roughly speaking, in a *public-key* ID scheme, a user registers its public key to a server. When the user wants to log in the server, the user proves its identity to the server by using a protocol. The security is captured by any polynomial-time adversary cannot impersonate the user. For the details of model and security notions, see Section 6.2.

Micciancio and Vadhan [MV03] proposed ID schemes based on lattice problems, such as GapSVP or GapCVP. These schemes are obtained from their statistical zero-knowledge protocol with efficient provers for the lattice problems. Lyubashevsky also constructed lattice-based ID schemes secure against active attack [Lyu08a]. Kawachi, Tanaka, and Xagawa [KTX08] proposed the ID schemes which are based on Stern's ID scheme [Ste96]. Finally, Lyubashevsky gave an efficient ID scheme based on the ideal-lattice-based hash functions [Lyu09].

These ID schemes (except the one of the Micciancio and Vadhan ID schemes) are secure against *concurrent* attack[1] under the assumptions on the *worst-case* hardness of lattice problems.

6.1.1 Main Ideas

In this section, we only discuss the ID schemes based on lattice problems rather than ideal-lattice-based ones, which is mainly same to the lattice-based one.

Quick remainder of the lattice-based hash family: We use the above relationship for our security reduction. Hence we mainly deals with SIS instead of GapSVP. Recall the lattice-based hash family $\mathcal{H}(q,m) = \mathsf{LHash}$ with a domain $D_n \subseteq \mathbb{Z}^m$. A key is a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. For $\boldsymbol{e} \in D_n$, a hash value is $h_{\boldsymbol{A}}(\boldsymbol{e}) := \boldsymbol{Ae} \bmod q$. Let d_2 be the maximum length of vectors in D_n. A collision $(\boldsymbol{e}, \boldsymbol{e}')$ of the hash function $h_{\boldsymbol{A}}$ implies a solution $\boldsymbol{z} = \boldsymbol{e} - \boldsymbol{e}'$ of $\mathrm{SIS}_{q,m,2d_2}$. Thus, the security of the hash family is based on the worst-case hardness of GapSVP with approximation factor $\tilde{O}(d_2 \cdot \sqrt{n})$ by Theorem 2.4.9.

Strategy to obtain concurrent security: The rough idea to obtain the concurrent security is summarized as follows: Fix the security parameter n and let $\mathcal{H}_n$ be a family of collision-resistant hash functions. Let h_a be the hash function with a key a. The secret key is $e \in D_n$ and the public key is $u = h_a(e)$. The prover proves its possession of e by a witness-indistinguishable and proof-of-knowledge (WIPoK) protocol. (The properties are defined later. See Section 6.2.) In a proof, a simulator simulates the prover oracle by using a secret key e. By using the knowledge extractor of the protocol, the simulator extracts a secret key e' such that $h_a(e') = u$. Then, it outputs e and e' as the collision of the hash functions. The witness indistinguishability ensures that $e \neq e'$ with certain probability.

Applying this strategy to the lattice-based hash functions, we can consider the following general construction: The public parameter is $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. The secret key is $\boldsymbol{e} \in D_n$ and the public key is $\boldsymbol{u} \leftarrow \boldsymbol{Ae} \bmod q$. The protocol is a WIPoK protocol for NP problem. The obtained scheme is less efficient because it employs the general WIPoK protocol.

[1] In *passive attack*, an adversary could only eavesdrop the transaction between the prover and the verifier. In *active attack*, an adversary could interact with the prover prior to impersonation. In *concurrent attack*, an adversary could interact with many different prover "clones" concurrently prior to impersonation. Each clone has the same secret key, but has independent random coins and maintains its own state. After interacting with many clones, the adversary tries impersonation. See the definition in Section 6.2.

To make a scheme efficient, the researchers tailored the protocols in the scheme. There are several difficulties to construct the protocol. We describe them and how to overcome them in each sections.

6.2 Definitions

In order to define models and security notions, we need to define protocols. In addition, we define properties of them.

6.2.1 Protocols

Provers and verifiers: An interactive algorithm A is a stateful algorithm that, given an incoming message M_{in} and state information st, outputs an outgoing message M_{out} and updated state st' (we will write $(M_{out}, st') \leftarrow \mathsf{A}(M_{in}, st)$). We say that A accepts if $st' = 1$ and rejects if $st' = 0$.

An interaction between a prover P and a verifier V ends when V either accepts or rejects. We will write

$$(tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(p_1, \dots)^{\mathrm{OP}_1,\dots} \leftrightarrow \mathsf{V}(v_1, \dots)^{\mathrm{OV}_1,\dots}]$$

to indicate that we let P having the accesses to the oracles $\mathrm{OP}_1, \dots$ interact with V having the accesses to the oracles $\mathrm{OV}_1, \dots$, having provided both P and V with fresh random coins, to get a transcript tr and a boolean decision dec.

Properties of Protocols

We first review the definition of a *view* of the verifier.

Definition 6.2.1. Let (P, V) be an interactive protocol. V's view of (P, V) on common input x, P's input w, V's input z is the random variable $\langle \mathsf{P}(w), \mathsf{V}(z) \rangle(x) = (r; m_1, \dots, m_t)$, where $m_1, \dots, m_t$ are exchanged messages between P and V and r is a random tape of V. That is, a random tape and a transcript between P and V.

We say an interactive protocol is an interactive proof system if the prover proves the validity of the instance x with the language L with completeness at least $2/3$ and soundness at least $1/3$.

Definition 6.2.2 (interactive proof system)**.** Let (P, V) be an interactive protocol. (P, V) is said to be an interactive proof system for a language L, if V is probabilistic polynomial-time algorithm and the followings hold:

1. For every $x \in L$,

$$\Pr_{\mathsf{P},\mathsf{V}}[dec = 1 : (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(x) \leftrightarrow \mathsf{V}(x)]] \geq 2/3.$$

2. For every $x \notin L$ and for every P^*,

$$\Pr_{\mathsf{P},\mathsf{V}}[dec = 1 : (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}^*(x) \leftrightarrow \mathsf{V}(x)]] \leq 1/3.$$

The quantities $2/3$ and $1/3$ can be replaced with c and $c - 1/\operatorname{poly}(n)$, where $c \in (0, 1)$ is a constant and n is the security parameter.

Zero knowledge: The zero-knowledge property captures the interaction to the prover P does not provide a knowledge with even a cheating verifier V^* *computationally*. (The interaction may give a knowledge to the verifier but this knowledge is useless for the polynomial-time algorithm V^*.) The idea is formulated by a simulator. If there is a simulator having no knowledge on witness and interacting with V^*, the provided knowledge is useless for V^*. We employ the black-box simulator definition for simplicity. See [Gol01, Section 4] for the details and the discussions on strength of definitions.

Definition 6.2.3 (black-box simulation zero knowledge)**.** We say an interactive proof system (P, V) for L is a perfect/statistical/computational-zero-knowledge protocol if there exists a probabilistic polynomial-time algorithm Sim such that

1. for all $x \in L$, $\Pr[\mathsf{Sim}^{\mathsf{V}^*}(x) = \perp] \leq 1/2$,
2. for every probabilistic polynomial-time V^* and for any $x \in L$

$$\widetilde{\mathsf{Sim}}^{\mathsf{V}^*}(1^n, x) \approx_{P/S/C} \langle \mathsf{P}, \mathsf{V} \rangle (1^n, x),$$

where $\widetilde{\mathsf{Sim}}^{\mathsf{V}^*}(s)$ denotes the output distribution of Sim having the oracle access to V^* on input s, conditioned on $\mathsf{Sim}(s) \neq \perp$.

Witness indistinguishability: Let L be an NP language, that is, there exist a polynomial $Q_L(\cdot)$ and a polynomial-time algorithm M_L such that,

1. For every $x \in L$, there exists $w \in \{0, 1\}^{Q_L(|x|)}$ such that $\mathsf{M}_L(x, w) = 1$.
2. For every $x \notin L$ and for any $w \in \{0, 1\}^{Q_L(|x|)}$, $\mathsf{M}_L(x, w) = 0$.

Then, we can define the binary relation $R_L = \{(x, w) | w \in \{0, 1\}^{Q_L(|x|)}$ such that $\mathsf{M}_L(x, w) = 1\}$. Suppose that x has two witnesses w and w' such that (x, w) and (x, w') in R_L. The witness indistinguishability says that the verifier cannot distinguish which witness the prover uses even if the verifier knows both witnesses. The formal definition is given below.

Definition 6.2.4. Let L be an NP language. Let (P, V) be an interactive proof system for L. We say that (P, V) is (perfectly/statistically/computationally) witness-indistinguishable if for every probabilistic verifier V^* running in time $\operatorname{poly}(n)$ and for any fixed $x \in L$ and $z \in \{0, 1\}^*$, for any two witnesses w_1 and w_2 for x

$$\langle \mathsf{P}(w_1), \mathsf{V}^*(z) \rangle (1^n, x) \approx_{P/S/C} \langle \mathsf{P}(w_2), \mathsf{V}^*(z) \rangle (1^n, x).$$

If the protocol is (perfectly/statistically/computationally) zero knowledge then the protocol is (perfectly/statistically/computationally) witness indistinguishable [FS90]. We omit the definition of witness hiding because we do not exploit this property explicitly. See [FS90] for the definition.

6.2.2 Model of Identification Schemes

We adopt the definition of identification schemes given in [AABN02]. An identification scheme SID is a quadruplet of algorithms $(\mathsf{Setup}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$.

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs public parameters *param*.

$\mathsf{KG}(param)$**:** A key-generation algorithm, given the public parameter *param*, outputs a key pair of a public key and a secret key (pk, sk).

$\mathsf{P}(param, pk, sk)$**,** $\mathsf{V}(param, pk)$**:** (P, V) is an interactive protocol. A prover algorithm P takes *param*, *pk*, and *sk* as inputs. A verifier algorithm V takes *param* and *pk* as inputs. At the end of interaction, V outputs 0 (reject) or 1 (accept).

We require the natural correctness condition; For any *param* and (pk, sk) generated by $\mathsf{Setup}(1^n)$ and $\mathsf{KG}(param)$, the decision of $\mathsf{V}(param, pk)$ interacting with $\mathsf{P}(param, pk, sk)$ is 1 with probability 1. That is,

$$\Pr\left[dec = 1 : \begin{array}{l} param \leftarrow \mathsf{Setup}(1^n); \\ (pk, sk) \leftarrow \mathsf{KG}(param, i); \\ (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(param, pk, sk) \leftrightarrow \mathsf{V}(param, pk)]; \end{array}\right] = 1.$$

An ID scheme is said to be *canonical* if the protocol is 3-move and public coin, that is,

$\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$**:** A prover algorithm consists of two algorithms P_1 and P_2.

$\mathsf{P}_1(param, pk, sk)$**:** A first prover algorithm, given *param*, *pk*, and *sk*, outputs a commitment *cmt* and its state information st_{P}.

$\mathsf{P}_2(ch, st_{\mathsf{P}})$**:** A second prover algorithm, given a challenge *ch* and a state information st_{P}, outputs a response *rsp*.

$\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$**:** A verifier algorithm consists of two algorithms V_1 and V_2.

$\mathsf{V}_1()$**:** A first verifier algorithm choose $ch \leftarrow C$ uniformly at random and outputs *ch*.

$\mathsf{V}_2(param, pk, cmt, ch, rsp)$**:** A second verifier algorithm, given *param*, *pk*, *cmt*, *ch*, and *rsp*, returns 0 (reject) or 1 (accept).

In the first move, the prover invokes P_1 and sends *cmt* to the verifier. In the second move, the verifier invokes V_1 with its randomness and sends *ch*, where this is the public coin since V_1 is the identity algorithm. In the third move, the prover invokes P_2 and sends *rsp* to the verifier.

6.2.3 Security Notions

We are interested in concurrent attack, which is stronger than active and passive attack. We employ the definition of concurrent security in [BP02]. In concurrent attack, the adversary will play the role of a cheating verifier prior to impersonation and can interact many different prover clones concurrently. Each clone has the same secret key, but has independent random coins and maintains its own state. We say SID is secure against impersonation under concurrent attack, if any polynomial-time adversary cannot, given a random public key of a legitimate prover, impersonate the legitimate prover.

We describe the formal definition as follows. Consider the experiment $\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{SID},\mathcal{A}}(n)$ between the challenger C and the impersonator $\mathcal{A} = (\mathsf{CV}, \mathsf{CP})$, where atk $\in \{\text{pa}, \text{aa}, \text{ca}\}$.

Experiment $\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{SID},\mathcal{A}}(n)$:

Setup Phase: The challenger C obtains $param \leftarrow \mathsf{Setup}(1^n)$. Next, C obtains $(pk, sk) \leftarrow \mathsf{KG}(param)$ and sets $PS \leftarrow \emptyset$, where PS denotes the set of prover's sessions. The impersonator CV is given the security parameter 1^n, the system parameter $param$, and the target public key pk.

Learning Phase: The impersonator CV can query to the prover oracle Prov.

- The oracle Prov receives inputs s, M_{in}. This oracle changes its behavior in three attacks.
  - If atk = pa, it obtains $(tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(param, pk, sk) \leftrightarrow \mathsf{V}(param, sk)]$ and returns (tr, dec) to the adversary.
  - If atk = aa, it runs as follows: If $s \notin PS$ then it sets $PS \leftarrow \{s\}$, picks a random coin ρ, and sets a state of the prover $st_{\mathsf{P}}[s] \leftarrow (param, sk, \rho)$. Next, it obtains $(M_{out}, st_{\mathsf{P}}[s]) \leftarrow \mathsf{P}(M_{in}, st_{\mathsf{P}}[s])$. It returns M_{out}.
  - If atk = ca, it runs as follows: If $s \notin PS$ then it adds s to PS (that is, $PS \leftarrow PS \cup \{s\}$), picks a random coin ρ, and sets a state of the prover $st_{\mathsf{P}}[s] \leftarrow (param, sk, \rho)$. Next, it obtains $(M_{out}, st_{\mathsf{P}}[s]) \leftarrow \mathsf{P}(M_{in}, st_{\mathsf{P}}[s])$. It returns M_{out}.

Challenge Phase: CV outputs st_{CP}. The challenger gives st_{CP} to CP. Finally, the challenger obtains $(tr, dec) \leftarrow \mathbf{Run}[\mathsf{CP}(st_{\mathsf{CP}}) \leftrightarrow \mathsf{V}(param, pk)]$ and returns dec.

Notice that if atk = pa the adversary could learn only transcripts between the legitimate prover and verifier. If atk = aa, the adversary could interact with the legitimate prover sequentially and has the power to abort the session. If atk = ca, the adversary interact with the legitimate prover concurrently by indicating each interaction with session identifier.

Definition 6.2.5. Let $\mathsf{SID} = (\mathsf{Setup}, \mathsf{KG}, \mathsf{P}, \mathsf{V})$ be an ID scheme, $\mathcal{A} = (\mathsf{CV}, \mathsf{CP})$ an impersonator, and n a security parameter. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\text{imp-atk}}_{\mathsf{SID},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{SID},\mathcal{A}}(n) = 1\right]$. We say that SID is secure against impersonation under passive, active, and concurrent attacks if $\mathbf{Adv}^{\text{imp-atk}}_{\mathsf{SID},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$ where atk = pa, aa, ca, respectively.

Special Soundness

We say a canonical ID scheme is special sound if an adversary, given *param* and *sk*, outputs (cmt, ch_1, rsp_1) and (cmt, ch_2, rsp_2) in the challenge phase with non-negligible probability such that $ch_1 \neq ch_2$ and $\mathsf{V}_2(param, pk, ch_1, rsp_1) = \mathsf{V}_2(param, pk, cmt, ch_2, rsp_2) = 1$ then we can compute *sk* corresponding to *pk*.

We say a canonical ID scheme is BS-special sound (in Bellare and Shoup [BS08]) if no polynomial-time adversary in the $\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{SID},\mathcal{A}}(n)$ cannot outputs (cmt, ch_1, rsp_1) and (cmt, ch_2, rsp_2) in the challenge phase with non-negligible probability such that $(ch_1, rsp_1) \neq (ch_2, rsp_2)$ and $\mathsf{V}_2(param, pk, ch_1, rsp_1) = \mathsf{V}_2(param, pk, cmt, ch_2, rsp_2) = 1$.

Often, the BS-special soundness is a stronger requirement than the special soundness.

6.3 The Micciancio–Vadhan Protocol

In [MV03], Micciancio and Vadhan proposed statistical zero-knowledge proof systems for GapCVP$_\delta$ and GapSVP$_\delta$. Here, we only discuss the one for GapCVP$_\delta$. Their protocol can be considered a zero-knowledge variant of the coAM protocol by Goldreich and Goldwasser [GG00].

Scheme 6.3.1 (The MV Protocol [MV03])**.** The protocol is parameterized by an integer k. The common input is a triplet $(\boldsymbol{B}, \boldsymbol{t}, d)$, which is an instance of GapCVP$_\delta$. Prover's auxiliary input is a lattice vector $\boldsymbol{Bw} \in \Lambda$ such that $\|\boldsymbol{t} - \boldsymbol{Bw}\| \leq d$. In the following, we denote $\boldsymbol{t} - \boldsymbol{Bw}$ by $\boldsymbol{u}$.

Step P1 (commitment):

1. For $i = 1, \ldots, k$, choose $c_i \in \{0, 1\}$ and $\boldsymbol{r}_i \in B(\delta d/2)$ uniformly at random.
2. Check that there exists an index i^* such that $\|\boldsymbol{r}_{i^*} + (2c_{i^*} - 1)\boldsymbol{u}\| \leq \delta d/2$ and store i^*. Otherwise, set $i^* = 1$ and redefine $c_{i^*} = 0$ and $\boldsymbol{r}_{i^*} = \boldsymbol{u}/2$ to satisfy $\|\boldsymbol{r}_{i^*} + (2c_{i^*} - 1)\boldsymbol{u}\| \leq \delta d/2$. (This procedure makes the protocol perfectly correct.)
3. Compute $\boldsymbol{y}_i = c_i\boldsymbol{t} + \boldsymbol{r}_i \bmod \boldsymbol{B}$ for all i.
4. Send $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_k$ to the verifier.

Step V1 (challenge): Flip a fair coin $c \leftarrow \{0, 1\}$ and send it to the prover.

Step P2 (response): Receive a bit $c \in \{0, 1\}$.

1. Compute $\boldsymbol{Bv}_i = \boldsymbol{y}_i - (\boldsymbol{r}_i + c_i\boldsymbol{t})$ for all i.

2. If $c \neq \bigoplus_i c_i$ then replace c_{i^*} and $\boldsymbol{B}\boldsymbol{v}_{i^*}$ by $1 - c_{i^*}$ and $\boldsymbol{B}(\boldsymbol{v}_{i^*} + (2c_{i^*} - 1)\boldsymbol{w})$.
3. Send $c_1, \dots, c_k$ and $\boldsymbol{v}_1, \dots, \boldsymbol{v}_k$ to the verifier.

Step V2 (verification): Receive k bits $c_1, \dots, c_k$ and k vectors $\boldsymbol{B}\boldsymbol{v}_1, \dots, \boldsymbol{B}\boldsymbol{v}_k \in L$. If $\bigoplus_i c_i = c$ and $\|\boldsymbol{y}_i - (\boldsymbol{B}\boldsymbol{v}_i + c_i\boldsymbol{t})\| \leq \delta d/2$ for all i the verifier accepts, otherwise rejects.

Intuitively, when the instance is an YES instance, the prover can *cheat* by $\boldsymbol{r}_{i^*}$ in both balls $B(\boldsymbol{0}, \delta d/2)$ and $B(\boldsymbol{t}, \delta d/2)$ since two balls overlap sufficiently. When the instance is a NO instance, two balls do not overlap and the prover cannot *cheat*. We note that the protocol is already k-concatenated by the ORing composition. Notice that we can simulate the prover oracle in passive attacks, since the protocol is honest-verifier statistical zero knowledge.

The properties of the protocol are summarized as follows:

Theorem 6.3.2 ([MV03, Lemma 4 and Corollary 6])**.** *Suppose that the security parameter is n, and $\Lambda(\boldsymbol{B}) \subseteq \mathbb{Z}^m$, where $m = \mathrm{poly}(n)$. The above system is a statistical zero-knowledge proof system for* GapCVP^2_δ *with perfect completeness and soundness error* $1/2$*, provided one of the following conditions holds true:*

- $\delta = \Omega(\sqrt{m/\log m})$ *and* $k = \mathrm{poly}(n)$ *is a sufficiently large polynomial,*
- *or* $\delta = \Omega(\sqrt{m})$ *and* $k = \omega(\log n)$ *is any super-logarithmic function of n,*
- *or* $\delta = m^{0.5+\Omega(1)}$ *and* $k = \omega(1)$ *is any super-constant function of n.*

Precisely speaking, there exists a simulator Sim *such that the statistical difference of* $\widetilde{\mathsf{Sim}}^{\mathsf{V}^*}(\boldsymbol{B}, \boldsymbol{t}, d)$ *from* $\langle \mathsf{P}, \mathsf{V}^* \rangle(\boldsymbol{B}, \boldsymbol{t}, d)$ *is at most* $2 \cdot (1 - \beta(2/\delta))^k$*, where* $\beta(\epsilon)$ *is the relative volume of the intersection of two m-dimensional unit spheres whose centers are at distance* ϵ*. Additionally, the protocol is honest-verifier statistical zero knowledge. Furthermore, there exists a knowledge extractor* KE*; if there exists a cheating prover* P^* *who makes* V *accept with probability* $1/2 + \epsilon$ *on some instance* $(\boldsymbol{B}, \boldsymbol{y}, d)$ *then* $\mathsf{KE}^{\mathsf{P}^*}(\boldsymbol{B}, \boldsymbol{t}, d)$ *outputs a lattice vector* $\boldsymbol{B}\boldsymbol{w} \in \mathcal{L}(\boldsymbol{B})$ *satisfying* $\|\boldsymbol{t} - \boldsymbol{B}\boldsymbol{w}\| \leq \delta d$ *in time* $\mathrm{poly}(n)/\epsilon^2$.

The above parameters are obtained by the bound $\beta(\epsilon) \geq \max\{3 \cdot \exp(-\epsilon^2 m/2), 1 - \epsilon\sqrt{m}\}$. See, for example, [GG00].

6.4 The Variants of the Micciancio–Vadhan Schemes

Combining the lattice-based hash family with the MV protocol [MV03], we obtain several ID schemes. In this section, we argue their concurrent security.

In [MV03, Section 5], they discussed identification schemes using their protocol. A summary of their discussions is as follows:

1. A passively secure $\omega(\log n)$-round ID scheme is obtained by sequential composition. The public key is an YES instance $(\boldsymbol{B}, \boldsymbol{t}, d)$ and the secret key is the corresponding witness $\boldsymbol{B}\boldsymbol{w}$.

2. A passively secure 3-round ID scheme is obtained by parallel composition. The public key is two YES instances of GapCVP_δ and the secret key is one of the corresponding witness.
3. A concurrently secure 3-round ID scheme is obtained by parallel composition and the ORing technique. The public key is two YES instances of GapCVP_δ and the secret key is one of the corresponding witness. (Applying techniques of De Santis, Di Crescenzo, Persiano, and Yung [DSDCPY94] and of Feige and Shamir [FS90], the ID scheme can be proven to have concurrent security.)
4. ($\mathsf{MV\text{-}ID}_{\mathsf{GL,p}}$) A passively secure 3-round ID scheme is obtained by using random lattices $\Lambda_q^\perp(\boldsymbol{A})$ under the worst-case assumptions of lattice problems. In order to obtain keys, one computes a public key $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and a secret key $\boldsymbol{e} \in \{0,1\}^m$ such that $\boldsymbol{A}\boldsymbol{e} \equiv \mathbf{0} \pmod{q}$. (See [Ajt96].) The prover, given a common input $(\boldsymbol{B}, \sqrt{m})$ and an auxiliary input $\boldsymbol{e}$, proves that $\boldsymbol{e} \in \Lambda_q^\perp(\boldsymbol{A})$ is short by using the MV protocol for GapSVP_δ^2.
5. ($\mathsf{MV\text{-}ID}^+_{\mathsf{GL,p}}$) A concurrently secure 3-round ID scheme is obtained by applying the ORing technique to the above passively secure ID scheme.
6. A passively secure 3-round ID scheme is obtained from the assumption that GapCVP with preprocessing (for the state-of-the-art hardness results of this problem, see [AKKV05]) is hard for some approximation factor. The third party chooses a common random matrix $\boldsymbol{B}$. Each user chooses a short error vector $\boldsymbol{x}$ as a secret key, and computes a public key $\boldsymbol{y} = \boldsymbol{x} \bmod \boldsymbol{B}$.

Their discussion (4) says that, by combining their protocol for GapSVP_δ and random lattices $\Lambda^\perp(\boldsymbol{A})$, we obtain an ID scheme which is secure against impersonation under passive attack under the worst-case hardness assumption of lattice problems. Their discussion (5) also says that we have a concurrently secure ID scheme based on the worst-case hardness of lattice problems[2].

An Observation: In more direct way, we obtain concurrently secure ID schemes ($\mathsf{MV\text{-}ID}^{++}_{\mathsf{L},*}$) by combining the lattice-based hash families $\mathcal{H}(q,m)$ or $\mathcal{H}(\mathbf{f},q,m)$ and the protocol for GapCVP_δ which are similar to the ID schemes in their discussion (6); The common lattice is set to be $\Lambda_q^\perp(\boldsymbol{A})$, where the third party publishes $\boldsymbol{A}$ uniformly chosen from $\mathbb{Z}_q^{n\times m}$. The secret key is $\boldsymbol{e}$ and the public key is $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \bmod q$.

A syndrome $\boldsymbol{u}$ indicates a target vector $\boldsymbol{t} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{t} \equiv \boldsymbol{u} \pmod{q}$. (See [GPV08, Section 5.1] for this isomorphism between a set of syndrome and that of

[2] In [Lyu08a, Section 1.2], Lyubashevsky wrote " In this work [MV03], the authors [Micciancio and Vadhan] show an efficient-prover SZK proof system for certain lattice problems and mention that one convert the proof system into an identification scheme. The conversion is non-trivial (due to the problem of zero-knowledge not being closed under parallel-composition), and many details remain to be filled in." But, it is easy to verify that the conversion yields a concurrently secure ID scheme, as they and we discussed in (5) $\mathsf{MV\text{-}ID}^+_{\mathsf{GL,p}}$. We note that the assumption is the worst-case hardness of $\mathrm{SIVP}_{\tilde{O}(n^{1.5})}$ and it is weaker than that of $\mathrm{SIVP}_{\tilde{O}(n^2)}$ Lyubashevsky used in [Lyu08a].

target vectors.) We can compute a close vector in lattice $\boldsymbol{x} = \boldsymbol{t} - \boldsymbol{e} \in \Lambda^{\perp}(\boldsymbol{A})$. The distance between $\boldsymbol{e}$ and $\boldsymbol{t}$ is exactly $\|\boldsymbol{x}\|_2$ which is at most d_2. So, the prover will run the MV protocol for a basis $\boldsymbol{B}$ of the lattice $\Lambda_q^{\perp}(\boldsymbol{A})$, a target vector $\boldsymbol{t}$, a threshold d, a parameter δ, and a secret vector $\boldsymbol{x}$.

The detail is in the next section.

6.4.1 Concrete Schemes

Let us describe the ID schemes, named $\mathsf{MV\text{-}ID}^{++}_{\mathsf{L},*}$, where $\mathsf{L} \in \{\mathsf{GL}, \mathsf{C/IL}\}$ denotes the underlying hash functions and $* \in \{p, s\}$ denotes parallel and sequential composition.

When we use ideal-lattice-based hash functions, we replace m with mn. We denote a set of keys of a hash family $\boldsymbol{A}$ by K_n, which is $\mathbb{Z}_q^{n\times m}$ in the case of the lattice-based hash family LHash and $\mathrm{Rot}_{\mathbf{f}}(R^m_{\mathbf{f},q})$ in the case of ideal-lattice-based hash family ILHash. Let d_2 and d_∞ denote $\max\{\|\boldsymbol{e}\|_2 : \boldsymbol{e} \in D_n\}$ and $\max\{\|\boldsymbol{e}\|_\infty : \boldsymbol{e} \in D_n\}$, respectively.

Scheme 6.4.1 ($\mathsf{MV\text{-}ID}^{++}_{\mathsf{L},*}$)**.** All of the participants agree the parameters m, q, $\mathbf{f}$, D_n, and $\delta = \Omega(\sqrt{n/\log n})$. The concrete scheme $\mathsf{MV\text{-}ID}^{++}_{\mathsf{L},*}$ is defined as follows.

$\mathsf{Setup}(1^n)$**:** Given the security parameter 1^n, the setup algorithm chooses $\boldsymbol{A} \leftarrow KS$, where $K_n = \mathbb{Z}_q^{n\times m}$ if $\mathsf{L} = \mathsf{GL}$ and $K_n = \mathrm{Rot}_{\mathbf{f}}(R^m_{\mathbf{f},q})$ if $\mathsf{L} = \mathsf{C/IL}$. In the following, $\boldsymbol{B}$ denotes a basis of $\Lambda_q^{\perp}(\boldsymbol{A})$ and all of the participants agree the matrix $\boldsymbol{B}$, say an Hermite normal form of the public lattice.

$\mathsf{KeyGen}(\mathsf{A})$**:** Given the public parameter $\boldsymbol{A}$, the key-generation algorithm chooses $\boldsymbol{e} \leftarrow D_n$ uniformly at random, computes $\boldsymbol{u} \leftarrow \boldsymbol{A}\boldsymbol{e} \bmod q$, and outputs $(pk, sk) \leftarrow (\boldsymbol{u}, \boldsymbol{e})$.

P **and** V**:** They interact as follows:

1. Compute a target vector $\boldsymbol{t} \in \mathbb{Z}^m$ or $\mathbb{Z}^{mn}$ such that $\boldsymbol{A}\boldsymbol{t} \equiv \boldsymbol{u} \pmod{q}$.
2. (The prover) compute $\boldsymbol{x} = \boldsymbol{t} - \boldsymbol{e}$.
3. If $\mathsf{L} = \mathsf{GL}$ set $d = d_2$. If $\mathsf{L} = \mathsf{C/IL}$ set $d = \sqrt{mn} \cdot d_\infty$.
4. They set $\boldsymbol{B}$, $\boldsymbol{t}$, and d as the common input and $\boldsymbol{x}$ as prover's auxiliary input. On the condition $* \in \{p, s\}$, they run the MV protocol for GapCVP_δ in parallel or sequential in $t = \omega(\log n)$ times, respectively.

The security of the protocol is summarized as follows:

Theorem 6.4.2. *Assume that $q^n / |D_n|$ is negligible in n. The above scheme $\mathsf{MV\text{-}ID}^{++}_{\mathsf{L},*}$ is concurrently secure where $\mathsf{L} = \mathsf{GL}$ or $\mathsf{C/IL}$ if $\mathrm{SIS}^2_{q,m,O(\delta d_2)}$ or $\mathbf{f}$-$\mathrm{SIS}^\infty_{q,m,O(\delta\sqrt{mn}d_\infty)}$ is hard on average. In particular,*

- *if $q = \mathrm{poly}(n)$, $m = \Theta(n \log q)$, $\delta = \sqrt{m}$, and $D_n = \{0,1\}^m$, then we have $d_2 = \sqrt{m}$ and the security of $\mathsf{MV\text{-}ID}^{++}_{\mathsf{GL},*}$ is based on the worst-case hardness of $\mathrm{SIVP}_{\tilde{O}(n^{1.5})}$, and*

- *if* $\mathbf{f}$ *is suitable,* $q = \mathrm{poly}(n)$*,* $m = \Theta(\log q)$*,* $\delta = \sqrt{mn}$*, and* $D_n = \{0,1\}^{mn}$*, then we have* $d_\infty = 1$ *and the security of* $\mathsf{MV\text{-}ID}^{++}_{\mathsf{C/IL},*}$ *is based on the worst-case hardness of* $\mathbf{f}\text{-SVP}^{\infty}_{\tilde{O}(n^2)}$.

In the proof, we use the simulator $\mathsf{MV.Sim}$ and the extractor $\mathsf{MV.KE}$ as the black box.

Proof. Since the MV protocol is witness indistinguishable, so are its parallel and sequential versions. The challenger, given a random matrix $\boldsymbol{A}$ from K_n, runs the adversary against concurrent security. The challenger makes a secret key $\boldsymbol{e}$ and a public key $\boldsymbol{u} \equiv \boldsymbol{A}\boldsymbol{e} \bmod q$. Using the secret key, it can simulate the prover oracle perfectly. Using the knowledge extractor $\mathsf{MV.KE}$, it obtains $\boldsymbol{x}' \in \Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\boldsymbol{t} - \boldsymbol{x}'\| \le \delta d$. Thus, if $\boldsymbol{x}'$ does not equal to $\boldsymbol{x} = \boldsymbol{t} - \boldsymbol{e}$, then we have a short vector $\boldsymbol{z} = \boldsymbol{x} - \boldsymbol{x}'$ in $\Lambda_q^{\perp}(\boldsymbol{A})$ whose length is at most $(\delta + 1)d$, since $\|\boldsymbol{x} - \boldsymbol{x}'\| \le \|\boldsymbol{t} - \boldsymbol{x}'\| + \|\boldsymbol{x} - \boldsymbol{t}\| \le \delta d + \|\boldsymbol{e}\|$. Next, we estimate the probability that $\boldsymbol{x} \neq \boldsymbol{x}'$. If $q^n / |D_n|$ is negligible in n, then a simple argument shows that, we have $\boldsymbol{x} \neq \boldsymbol{x}'$ with probability at least $1/2$, since the MV protocol is witness indistinguishable.

If $\mathsf{L} = \mathsf{GL}$, d is set as d_2. Thus, the length of $\boldsymbol{z}$ is at most $(\delta + 1)d = O(\delta d_2)$. If $\mathsf{L} = \mathsf{C/IL}$, the threshold d is set as $\sqrt{mn} \cdot d_\infty$. Hence, the max norm of $\boldsymbol{z}$ is at most $(\delta + 1)d = O(\delta\sqrt{mn} \cdot d_\infty)$. This completes the proof. □

6.5 Lyubashevsky's Scheme – 1

We next review the Lyubashevsky ID schemes $\mathsf{Ly08\text{-}ID}_{\mathsf{L},p}$ [Lyu08a], where $\mathsf{L} \in \{\mathsf{GL}, \mathsf{C/IL}\}$.

The protocol is algebraic structure, while the MV protocol exploited the geometric structure.

Let us recall the Random-or-Masked protocol often used in the protocols for the number-theoretic relations. The typical example is the Schnorr protocol [Sch91]. Let g be a generator of a cyclic group $\mathbb{G}$ of order prime q. Let a common input be $(g, \mathbb{G}, q, u = g^e)$ and auxiliary input $e \in \mathbb{Z}_q$. In the protocol, (1) the prover chooses $r \leftarrow \mathbb{Z}_q$ and commits $y = g^r$, (2) the verifier chooses a challenge $c \leftarrow \{0,1\}$, which corresponds verifier's order to open "random" or "masked" values, (3) the prover responds $z = ce + r \bmod q$, and (4) the verifier accepts if $z \in [0, q-1]$ and $g^z = u^c \cdot y$. The prover opens r if $c = 1$ and it opens $e + r$ otherwise. It is easy to show that the protocol has soundness $1/2$ and is perfectly zero knowledge and proof of knowledge (in addition, has special soundness).

Lyubashevsky applied this strategy to the lattice-based hash functions. The auxiliary input is $\boldsymbol{e} \leftarrow D_n = \{0,1\}^m$ and the common input is $\boldsymbol{A}$ and $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \bmod q$. In the first attempt, the protocol is (1) the prover chooses $\boldsymbol{r} \in [0, \dots, D]^m$ and commits $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{r} \bmod q$, (2) the verifier chooses a challenge $c \leftarrow \{0,1\}$, (3) the prover responds $\boldsymbol{z} = c\boldsymbol{e} + \boldsymbol{r}$, and (4) the verifier accepts if $\boldsymbol{z} \in [0, \dots, D+1]^m$ and $\boldsymbol{A}\boldsymbol{z} \equiv c\boldsymbol{u} + \boldsymbol{y} \pmod{q}$.

But, this direct approach fails, facing a dilemma. If we set $D = q - 1$, the adversary without knowledge of $\boldsymbol{e}$ can make the verifier accept. On the contrary, if we set $D < q - 1$, the response $\boldsymbol{z}$ will leak the secret since $q = \mathrm{poly}(n)$. If $e_1 = 1$, the first coordinate of the response z_1 takes a value $D+1$ with probability $1/(D+1)$, while z_1 cannot takes a value $D + 1$ if $e_1 = 0$.

This problem is overcome by the discard of the response. The prover aborts the protocol if $\boldsymbol{z}$ leaks the secret $\boldsymbol{e}$. The abortion makes the protocol not zero knowledge, but we can show the protocol is witness indistinguishable by taking the parameters carefully.

The basic protocol is defined as follows:

Scheme 6.5.1 (Basic protocol [Lyu08a])**.** All of the participants agree with the parameter $m = m(n)$ and $q = q(n)$. In addition, they agree with the sets D_e, D_r, and G.

$\mathsf{Setup}(1^n)$**:** The setup algorithm, given 1^n, outputs a random matrix $\boldsymbol{A} \leftarrow K_n$.

$\mathsf{KeyGen}(\boldsymbol{A})$**:** The key-generation algorithm, given the public parameter $\boldsymbol{A}$, chooses a random vector $\boldsymbol{e} \in D_e$ and computes $\boldsymbol{u} \leftarrow h_{\boldsymbol{A}}(\boldsymbol{e}) \in \mathbb{Z}_q^n$. It outputs $(pk, sk) = (\boldsymbol{u}, \boldsymbol{e})$.

$\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$, $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$**:** The common inputs are $\boldsymbol{A}$ and $\boldsymbol{u}$. Prover's auxiliary input is $\boldsymbol{e}$. They interact as follows:

Step P1: Pick a random $\boldsymbol{r} \leftarrow D_r$ and send $\boldsymbol{y} \leftarrow h_{\boldsymbol{A}}(\boldsymbol{r})$.

Step V1: Send a random challenge $c \leftarrow \{0, 1\}$.

Step P2: Compute $\boldsymbol{z} \leftarrow c\boldsymbol{e} + \boldsymbol{r}$. If $\boldsymbol{z} \in G$, then send it to the verifier. Otherwise, send $\perp$ and abort the protocol.

Step V2: Receiving $\boldsymbol{z}$, accepts if $\boldsymbol{z} \in G$ and $h_{\boldsymbol{A}}(\boldsymbol{z}) = c\boldsymbol{u} + \boldsymbol{y}$.

In the following, we only discuss the case where $\mathsf{L} = \mathsf{GL}$. The choice of the parameters is as follows: $m = \lfloor 4n \log n \rfloor$, $q = \widetilde{\Theta}(n^3)$, $D_e = \{0, 1\}^m$, $D_r = \{0, 1, \dots, 5m - 1\}^m$, and $G = [5m - 1]^m$.[3]

Lyubashevsky showed the followings:

1. For $m \geq 10$, the completeness error is at most 0.19, that is, $\Pr_{\mathsf{P},\mathsf{V}}[dec = 1 : (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{e}) \leftrightarrow \mathsf{V}(\boldsymbol{A}, \boldsymbol{u})]] \geq (1 - 1/5m)^m \geq 0.81$.
2. The protocol is statistically witness indistinguishable.
3. For any $\boldsymbol{A}$, $\Pr_{\boldsymbol{e} \leftarrow \{0,1\}^m}[\exists \boldsymbol{e}' \in \{0, 1\}^m \setminus \{\boldsymbol{e}\}, h_{\boldsymbol{A}}(\boldsymbol{e}) = h_{\boldsymbol{A}}(\boldsymbol{e}')] \geq 1 - 2^{n \log q - m}$.

For the proofs, see [Lyu08a]. In the next section, we give the full description of the scheme $\mathsf{Ly08\text{-}ID}_{\mathsf{L},*}$.

[3] we change the verification procedure. In the original, the verifier checks $\|z\| \leq 5m^{1.5}$ instead of $z \in G$.

6.5.1 Description

Scheme 6.5.2 ($\mathsf{Ly08\text{-}ID}_{\mathsf{L},p}$ [Lyu08a]). All of the participants agree with the parameter $m = m(n)$ and $q = q(n)$. In addition, they agree with the sets D_e, D_r, and G.

$\mathsf{Setup}(1^n)$**:** The setup algorithm, given 1^n, outputs a random matrix $\boldsymbol{A} \leftarrow K_n$.

$\mathsf{KeyGen}(\boldsymbol{A})$**:** The key-generation algorithm, given the public parameter $\boldsymbol{A}$, chooses a random vector $\boldsymbol{e} \in D_e$ and computes $\boldsymbol{u} \leftarrow h_{\boldsymbol{A}}(\boldsymbol{e}) \in \mathbb{Z}_q^n$. It outputs $(pk, sk) = (\boldsymbol{u}, \boldsymbol{e})$.

$\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$**,** $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$**:** The common inputs are $\boldsymbol{A}$ and $\boldsymbol{u}$. Prover's auxiliary input is $\boldsymbol{e}$. The protocol is t-parallel or t-sequential composition of the basic protocol. If the verifier of the basic protocol accepts at least 0.65 fraction of the t protocols, then the verifier accepts. Otherwise, it rejects.

The completeness error is reduced to at most $2^{-t/14}$, shown by the Chernoff bound. The security of $\mathsf{Ly08\text{-}ID}_{\mathsf{GL},p}$ is summarized as follows:

Theorem 6.5.3 (Theorem 13, [Lyu08a])**.** *If there is an adversary breaking* $\mathsf{Ly08\text{-}ID}_{\mathsf{GL},p}$ *in time T and with probability ϵ, then there exists an algorithm solving* $\mathrm{SIS}_{q,m,\beta}$ *in time* $\mathrm{poly}(T, n)$ *with success probability* $\Omega(\epsilon^2 - 2^{-t/18+1}) - \mathsf{negl}(n)$*, where* $\beta = 10m^{1.5}$.

We omit the proofs, see the original paper [Lyu08a]. We note that the proof for the ideal-lattice-based scheme $\mathsf{Ly08\text{-}ID}_{\mathsf{C/IL},p}$ is obtained in the similar way to the above.

6.6 Review of Stern's ID Scheme

Here, we turn our eyes to the identification schemes based on coding problems. The Stern ID scheme is the first one based on the hardness of the coding problems. Stern's protocol deals with the decoding problem on binary codewords called the Syndrome Decoding Problem.

Definition 6.6.1 (Syndrome Decoding Problem)**.** Given $\boldsymbol{H} \in \mathbb{Z}_2^{n\times m}$, $\boldsymbol{u} \in \mathbb{Z}_2^n$, and $w \in \mathbb{N}$, the problem is finding a vector $\boldsymbol{e} \in \mathcal{S}(m, w)$ such that $\boldsymbol{He} \equiv \boldsymbol{u} \bmod 2$.

We can consider this problem as a restricted version of $\mathrm{ISIS}_{q,m,\beta}$ (by replacing $\boldsymbol{H}$ with $\boldsymbol{A}$ and 2 with q). He indeed proposed that an analogous scheme in $\mathbb{Z}_q$, where q is extremely small (typically 3, 5, or 7) [Ste96, Section VI].

Let us consider the protocol, where the common input is $\boldsymbol{H} \in \mathbb{Z}_q^{n\times m}$, $\boldsymbol{u} = \boldsymbol{He} \bmod q$, and the *Hamming weight* w of $\boldsymbol{e}$. Prover's auxiliary input is $\boldsymbol{e}$. Stern's protocol is a Random–Masked–Permute protocol, which allows the prover to prove the Hamming weight of the auxiliary input $\boldsymbol{e}$. (1) the prover commits a masked value $\boldsymbol{Hr}$, a permutation π, and permuted vectors $\pi(\boldsymbol{e})$ and $\pi(\boldsymbol{r})$, (2) the verifier chooses a challenge $c \leftarrow \{1, 2, 3\}$ corresponding to the order opening "permuted," "masked," and "random" values, (3) the prover opens $(\pi(\boldsymbol{e}), \pi(\boldsymbol{r}))$, $(\pi, \boldsymbol{e} + \boldsymbol{r})$, or $(\pi, \boldsymbol{r})$, (4) the verifier accepts if the checks are passed. Notice that the verifier can verify the Hamming weight of $\boldsymbol{e}$ in the check of the permuted values.

The precise protocols is given below:

Scheme 6.6.2 (The basic scheme in [Ste96])**.** All of the participants agree with the parameter $m = m(n)$, $q = q(n)$, and the weight $w = w(n)$. They also agree the hash function $H : \{0,1\}^{2k} \to \{0,1\}^{l}$. Let us define the commitment function $Com : \{0,1\}^{2k} \to \{0,1\}^{l}$ as $Com(msg; \rho) = H(\rho \circ (msg \oplus \rho))$ for $msg, \rho \in \{0,1\}^{k}$. We omit the randomness part ρ in the description.

Setup(1^n)**:** The setup algorithm, on input 1^n, outputs a random matrix $\boldsymbol{H} \in \mathbb{Z}_q^{n \times m}$.

KG($\boldsymbol{H}$)**:** The key-generation algorithm, on input $\boldsymbol{H}$, chooses a random vector $\boldsymbol{e} \in \mathcal{S}(m, w)$ and computes $\boldsymbol{u} := \boldsymbol{He} \bmod q$. It outputs $(pk, sk) = (\boldsymbol{u}, \boldsymbol{e})$.

$\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$, $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$**:** The common inputs are $\boldsymbol{H}$ and $\boldsymbol{u}$. Prover's auxiliary input is $\boldsymbol{e}$. They interact as follows:

Step P1: Choose a random permutation π over $[m]$ and a random vector $\boldsymbol{r} \in \mathbb{Z}_q^m$ and send commitments c_1, c_2, and c_3 computed as

- $c_1 = Com(\pi, \boldsymbol{Hr})$,
- $c_2 = Com(\pi(\boldsymbol{r}))$,
- $c_3 = Com(\pi(\boldsymbol{e} + \boldsymbol{r}))$.

Step V1: Send a random challenge $ch \in \{1, 2, 3\}$ to P.

Step P2:

- If $ch = 1$, reveal c_2 and c_3. So, send $\boldsymbol{w} = \pi(\boldsymbol{e})$ and $\boldsymbol{x} = \pi(\boldsymbol{r})$.
- If $ch = 2$, reveal c_1 and c_3. Send $\phi = \pi$ and $\boldsymbol{y} = \boldsymbol{e} + \boldsymbol{r}$.
- If $ch = 3$, reveal c_1 and c_2. Send $\psi = \pi$ and $\boldsymbol{z} = \boldsymbol{r}$.

Step V2:

- If $ch = 1$, check that $c_2 = Com(\boldsymbol{x})$, $c_3 = Com(\boldsymbol{w} + \boldsymbol{x})$, and $\boldsymbol{w} \in \mathcal{S}(m, w)$.
- If $ch = 2$, check that $c_1 = Com(\phi, \boldsymbol{Hy} - \boldsymbol{u})$ and $c_3 = Com(\phi(\boldsymbol{y}))$.
- If $ch = 3$, check that $c_1 = Com(\psi, \boldsymbol{Hz})$ and $c_2 = Com(\psi(\boldsymbol{z}))$.

Output $dec = 1$ if all checks are passed, otherwise output $dec = 0$.

In [Ste96], Stern insisted that the protocol is SZKPoK protocol and yields the passively secure ID scheme based on the average-case hardness of the syndrome decoding problem, where $\boldsymbol{H}$ and $\boldsymbol{u}$ are uniformly at random over $\mathbb{Z}_q^{n \times m}$ and $\mathbb{Z}_q^n$. However, we could not prove the security of the commitment function despite of our efforts. We can show the security if we replace the above commitment with the statistically-hiding and computationally-binding commitment. We omit the proof, since it is given in the next section.

6.7 The Kawachi–Tanaka–Xagawa Identification Scheme

Kawachi et al. [KTX08] observed that the key-generation algorithm of the above basic scheme has a similar structure of the lattice-based hash functions LHash. In addition, they also observed that, if the commitment *Com* is replaced with LNIC, the underlying problems are now $\mathrm{SIS}_{q,m,\beta}$ for an appropriate β.

Following their observations, let us replace $\boldsymbol{H}$ with $\boldsymbol{A}$ and *Com* with LNIC, i.e., $Com_{\boldsymbol{A}}$, in Section 5.3 in the above basic scheme, Scheme 6.6.2. Then the following reduction algorithm shows the concurrent security: On input $\boldsymbol{A}$, generates a secret key $\boldsymbol{e} \in \mathcal{S}(m, w)$ and a public key $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \bmod q$, and feeds $\boldsymbol{A}$ and $\boldsymbol{u}$ to the adversary. The reduction algorithm can simulate the prover that the adversary concurrently accesses, since the algorithm has $\boldsymbol{A}$ and $\boldsymbol{e}$. Using the knowledge extractor for the adversary in Stern's proof, the algorithm obtains either a collision of a string commitment scheme or a secret key $\boldsymbol{e}'$ such that $\boldsymbol{e}' \neq \boldsymbol{e}$ and $\boldsymbol{A}\boldsymbol{e}' = \boldsymbol{u}$. In the former case, the algorithm outputs the collision $(\boldsymbol{s}, \boldsymbol{s}')$ of a hash function $h_{\boldsymbol{A}}$ in the string commitment scheme. Thus, the solution for SIS is obtained by $\boldsymbol{z} = \boldsymbol{s} - \boldsymbol{s}'$. In the latter case, the condition $\boldsymbol{e} \neq \boldsymbol{e}'$ will be satisfied with probability at least $1/2$ by witness indistinguishability of Stern's protocol. Thus, the algorithm has the solution $\boldsymbol{z} = \boldsymbol{e} - \boldsymbol{e}'$ for SIS. The l_2 norm of both solutions is at most $\sqrt{m} = \tilde{O}(n^{1/2})$. From the relationship between SIS and GapSVP the assumption is the worst-case hardness of $\mathrm{GapSVP}_{\tilde{O}(n)}$.

6.7.1 Description

The variant $\mathsf{St\text{-}ID}^{+}_{\mathsf{GL},*}$ (for $\mathsf{L} \in \{\mathsf{GL}, \mathsf{C/IL}\}$ and $* \in \{p, s\}$) is obtained by replacing the string commitment scheme in Stern's ID scheme [Ste96] with our lattice-based one. We adjust this parameter to connect his framework to our assumptions of the lattice problems.

We now describe the protocol $\mathsf{St\text{-}ID}^{+}_{\mathsf{GL},*}$ below. To simplify the notations, we do not write random strings in $Com_{\boldsymbol{A}}$ explicitly.

Scheme 6.7.1 ($\mathsf{St\text{-}ID}^{+}_{\mathsf{GL},*}$, [KTX08])**.** If $* = s$ the protocol is repeated sequentially t times. If $* = p$ the protocol is composed in t parallel sessions.

$\mathsf{Setup}(1^n)$**:** The setup algorithm, on input 1^n, outputs a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$. Notice that this matrix defines the hash function $h_{\boldsymbol{A}}$ and the commitment function $Com_{\boldsymbol{A}}$.

$\mathsf{KG}(A)$**:** The key-generation algorithm, on input $\boldsymbol{A}$, chooses a random vector $\boldsymbol{e} \in \mathcal{S}(m, w)$ and computes $\boldsymbol{u} := \boldsymbol{A}\boldsymbol{e} \bmod q$. It outputs $(pk, sk) = (\boldsymbol{u}, \boldsymbol{e})$.

$\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$**,** $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$**:** The common inputs are $\boldsymbol{A}$ and $\boldsymbol{u}$. Prover's auxiliary input is $\boldsymbol{e}$. The verifier accepts if all verifiers accept.

Step P1: Choose a random permutation π over $[m]$ and a random vector $\boldsymbol{r} \in \mathbb{Z}_q^m$ and send commitments c_1, c_2, and c_3 computed as

- $c_1 = Com_{\boldsymbol{A}}(\pi, \boldsymbol{A}\boldsymbol{r})$,

- $c_2 = Com_{\boldsymbol{A}}(\pi(\boldsymbol{r}))$,
- $c_3 = Com_{\boldsymbol{A}}(\pi(\boldsymbol{e} + \boldsymbol{r}))$.

Step V1: Send a random challenge $ch \in \{1, 2, 3\}$ to P.

Step P2:

- If $ch = 1$, reveal c_2 and c_3. So, send $\boldsymbol{w} = \pi(\boldsymbol{e})$ and $\boldsymbol{x} = \pi(\boldsymbol{r})$.
- If $ch = 2$, reveal c_1 and c_3. Send $\phi = \pi$ and $\boldsymbol{y} = \boldsymbol{e} + \boldsymbol{r}$.
- If $ch = 3$, reveal c_1 and c_2. Send $\psi = \pi$ and $\boldsymbol{z} = \boldsymbol{r}$.

Step V2:

- If $ch = 1$, check that $c_2 = Com_{\boldsymbol{A}}(\boldsymbol{x})$, $c_3 = Com_{\boldsymbol{A}}(\boldsymbol{w} + \boldsymbol{x})$, and $\boldsymbol{w} \in \mathcal{S}(m, m/2)$.
- If $ch = 2$, check that $c_1 = Com_{\boldsymbol{A}}(\phi, \boldsymbol{A}\boldsymbol{y} - \boldsymbol{u})$ and $c_3 = Com_{\boldsymbol{A}}(\phi(\boldsymbol{y}))$.
- If $ch = 3$, check that $c_1 = Com_{\boldsymbol{A}}(\psi, \boldsymbol{A}\boldsymbol{z})$ and $c_2 = Com_{\boldsymbol{A}}(\psi(\boldsymbol{z}))$.

Output $dec = 1$ if all checks are passed, otherwise output $dec = 0$.

6.7.2 Security Proofs

We will show the followings and prove the security by composing them.

1. The completeness error is 0.
2. The protocol (P, V) is an SZK protocol and thus it is statistically witness-indistinguishable.
3. $\Pr_{\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}, \boldsymbol{e} \leftarrow \mathcal{S}(m,w)}[\exists\, \boldsymbol{e}' \in \mathcal{S}(m, w), h_{\boldsymbol{A}}(\boldsymbol{e}) = h_{\boldsymbol{A}}(\boldsymbol{e}')] \geq 1 - \mathsf{negl}(n)$.
4. There is a knowledge extractor KE extracting a collision in $h_{\boldsymbol{A}}$ from an adversary.

The first part is easily verified.

Next, we show that the protocol is an SZK protocol. The proof of zero-knowledge property of the original protocol is in [Ste96, Theorem 4]. Stern left completion of the proof as the problem for reader. Thus, we give the whole proof that Stern's protocol is statistically zero knowledge when *Com* is a statistically-hiding and computationally-binding string commitment scheme.

Lemma 6.7.2. *The protocol is statistically zero knowledge when Com is a statistically-hiding and computationally-binding string commitment scheme.*

Proof. Following the definition, we construct a simulator $\mathcal{S}$ which on input $\boldsymbol{A}$ and $\boldsymbol{y}$ and given oracle access to a cheating verifier CV, outputs a simulated transcript. A real transcript between P and CV on input $\boldsymbol{A}$ and $\boldsymbol{y}$ is denoted by $\langle \mathsf{P}, \mathsf{CV} \rangle(\boldsymbol{A}, \boldsymbol{y})$.

First, $\mathcal{S}$ chooses a random value $\bar{c}$ from $\{1, 2, 3\}$ which is a prediction what value the cheating verifier CV will *not* choose. Next, it chooses a random tape of CV, denoted by r'. We remark that, by the assumption on the commitment, the distributions of a challenge from CV in the real interaction and that in the simulation are statistically close.

Case $\bar{c} = 1$: $\mathcal{S}$ computes $\boldsymbol{x}' \in \mathbb{Z}_q^m$ such that $\boldsymbol{A}\boldsymbol{x}' = \boldsymbol{y}$ by using linear algebra. Next, it chooses a random permutation π' over $[m]$, a random vector $\boldsymbol{r}' \in \mathbb{Z}_q^m$, and random strings ρ_1', ρ_2', and ρ_3'. So, it computes

- $c_1' := \mathit{Com}(\pi', \boldsymbol{A}\boldsymbol{r}'; \rho_1')$,
- $c_2' := \mathit{Com}(\pi'(\boldsymbol{r}'); \rho_2')$,
- $c_3' := \mathit{Com}(\pi'(\boldsymbol{x}' + \boldsymbol{r}'); \rho_3')$.

It sends them to CV. Since the commitment scheme is statistically hiding, the distribution of a challenge from CV is statistically close to the real distribution. Receiving a challenge ch from CV, the simulator $\mathcal{S}$ computes a transcript as follows:

- If $ch = 1$, $\mathcal{S}$ outputs $\perp$ and halts.
- If $ch = 2$, it outputs $(r'; (c_1', c_2', c_3'), 2, (\pi', \boldsymbol{x}' + \boldsymbol{r}', \rho_1', \rho_3'))$.
- If $ch = 3$, it outputs $(r'; (c_1', c_2', c_3'), 3, (\pi', \boldsymbol{r}', \rho_1', \rho_2'))$.

We analyze the case $ch = 2$. In this case, we obtain that

$$\begin{aligned} \langle \mathsf{P}, \mathsf{CV} \rangle(\boldsymbol{A}, \boldsymbol{y}) &= (r; (c_1, c_2, c_3), 2, (\pi, \boldsymbol{x} + \boldsymbol{r}, \rho_1, \rho_3), \\ \mathcal{S}(\boldsymbol{A}, \boldsymbol{y}) &= (r'; (c_1', c_2', c_3'), 2, (\pi', \boldsymbol{x}' + \boldsymbol{r}', \rho_1', \rho_3')). \end{aligned}$$

Assume that $(\pi', \boldsymbol{r}', \rho_1', \rho_3') = (\pi, \boldsymbol{r} + \boldsymbol{x} - \boldsymbol{x}', \rho_1, \rho_3)$. By this equation, we have that $c_1' = c_1$, $c_3' = c_3$, and the responses from the simulator equal to the responses from the prover. Since the commitment is statistically hiding, we have the distributions of c_2 and c_2' are statistically close. Thus, we conclude that the both distributions of the simulated transcript and the real transcript are statistically close.

It is straightforward to show it in the case $ch = 3$ by using the equation $(\pi', \boldsymbol{r}') = (\pi, \boldsymbol{r})$. Thus, we omit this part from the proof.

Case $\bar{c} = 2$: $\mathcal{S}$ chooses a random permutation π' over $[m]$, two random vectors $\boldsymbol{r}' \in \mathbb{Z}_q^m$, $\boldsymbol{x}' \in \mathcal{S}(m, m/2)$, and random strings ρ_1', ρ_2', and ρ_3'. $\mathcal{S}$ computes commitments

- $c_1' := \mathit{Com}(\pi', \boldsymbol{A}\boldsymbol{r}'; \rho_1')$,
- $c_2' := \mathit{Com}(\pi'(\boldsymbol{r}'); \rho_2')$,
- $c_3' := \mathit{Com}(\pi'(\boldsymbol{x}' + \boldsymbol{r}'); \rho_3')$.

It sends them to CV. Receiving a challenge ch, the simulator computes a transcript as follows:

- If $ch = 1$, then $\mathcal{S}$ outputs $(r'; (c_1', c_2', c_3'), 1, (\pi'(\boldsymbol{x}'), \pi'(\boldsymbol{r}'), \rho_2', \rho_3'))$.
- If $ch = 2$, then it outputs $\perp$ and halts.
- If $ch = 3$, then it outputs $(r'; (c_1', c_2', c_3'), 3, (\pi', \boldsymbol{r}', \rho_1', \rho_2'))$.

We analyze the case $ch = 1$. In this case, we have that

$$\begin{aligned}\langle \mathsf{P}, \mathsf{CV}\rangle(\boldsymbol{A}, \boldsymbol{y}) &= (r; (c_1, c_2, c_3), 1, (\pi(\boldsymbol{x}), \pi(\boldsymbol{r}), \rho_2, \rho_3), \\ \mathcal{S}(\boldsymbol{A}, \boldsymbol{y}) &= (r'; (c_1', c_2', c_3'), 1, (\pi'(\boldsymbol{x}'), \pi'(\boldsymbol{r}'), \rho_2', \rho_3')).\end{aligned}$$

Let χ be a permutation over $[m]$ such that $\chi(\boldsymbol{x}') = \boldsymbol{x}$. In this case, we set $(\pi', \boldsymbol{r}', \rho_2', \rho_3') = (\chi^{-1} \circ \pi, \chi(\boldsymbol{r}), \rho_2, \rho_3)$. By this equation, we have $c_2' = c_2$, $c_3' = c_3$, and the responses from the simulator equal to the responses from the prover. Since the commitment scheme is statistically hiding, the distributions of the real transcript and the output of the simulator are statistically close.

We omit the proof of the case $ch = 3$, since it is trivial.

Case $\bar{c} = 3$: $\mathcal{S}$ chooses a random permutation π over $[m]$, two random vectors $\boldsymbol{r} \in \mathbb{Z}_q^m$, $\boldsymbol{x}' \in \mathcal{S}(m, m/2)$, and random strings ρ_1, ρ_2, and ρ_3. $\mathcal{S}$ computes

- $c_1 := Com(\pi, \boldsymbol{A}(\boldsymbol{x}' + \boldsymbol{r}) - \boldsymbol{y}; \rho_1)$,
- $c_2 := Com(\pi(\boldsymbol{r}); \rho_2)$,
- $c_3 := Com(\pi(\boldsymbol{x}' + \boldsymbol{r}); \rho_3)$.

It sends them to CV.

- If $ch = 1$, then $\mathcal{S}$ outputs $(r'; (c_1, c_2, c_3), 1, (\pi(\boldsymbol{x}'), \pi(\boldsymbol{r}), \rho_2, \rho_3)$.
- If $ch = 2$, then it outputs $(r'; (c_1, c_2, c_3), 2, (\pi, \boldsymbol{x}' + \boldsymbol{r}'))$.
- If $ch = 3$, it outputs $\perp$ and halts.

In the case $ch = 1$, we consider the equation $(\pi', \boldsymbol{r}', \rho_2', \rho_3') = (\chi^{-1} \circ \pi, \chi(\boldsymbol{r}), \rho_2, \rho_3)$. The remaining part of proof is the same as that in the case $\bar{c} = 2$ and $ch = 1$. In the case $ch = 2$, we let $(\pi', \boldsymbol{r}', \rho_1', \rho_3') = (\pi, \boldsymbol{r} + \boldsymbol{x} - \boldsymbol{x}', \rho_1, \rho_3)$. The remaining part of proof is the same as that in the case $\bar{c} = 1$ and $ch = 2$.

The probability that the simulator $\mathcal{S}$ outputs $\perp$ is at most $1/3 + \epsilon(n) \leq 1/2$ where ϵ is some negligible function. Additionally, by the above arguments, the distribution of the output of $\mathcal{S}$ conditioned on it is not $\perp$ is statistically close to the distribution of the real transcript. Therefore, we have constructed the simulator and completed the proof. □

Since the protocol is statistically zero knowledge for $t = 1$, it has a witness-indistinguishable property. Witness-indistinguishable property is closed under the parallel composition [FS90]. Thus, the above protocol is witness indistinguishable for $t = \omega(\log n)$ if a statistically-hiding string commitment scheme is used.

We show the theorem of the security on our ID protocol, which concerns impersonation under concurrent attack.

Theorem 6.7.3. *For any $q = \mathrm{poly}(n)$, $m \geq 2(1 + \delta)n \log q$ for some constant $\delta > 0$, and $w = \omega(\log m)$ such that $q^n / |\mathcal{S}(m, w)|$ is negligible in n, the above ID scheme* $\mathsf{St\text{-}ID}^{+}_{\mathsf{GL},*}$ *is concurrently secure if* $\mathrm{SIS}_{q,m,\sqrt{m}}$ *is hard on average.*

Before the proof of security, we need to mention the following trivial lemma, which corresponds to the third part.

Lemma 6.7.4. *For any fixed* $\boldsymbol{A}$*, let* $U := \{\boldsymbol{u} \in \mathbb{Z}_q^n \mid |\{\boldsymbol{e} \in \mathcal{S}(m,w) \mid \boldsymbol{A}\boldsymbol{e} = \boldsymbol{u}\}| = 1\}$*, i.e., a set of vectors* $\boldsymbol{u}$ *such that the preimage* $\boldsymbol{e}$ *of* $\boldsymbol{u}$ *is uniquely determined for* $\boldsymbol{A}$*. If* $q^n / |\mathcal{S}(m,w)|$ *is negligible in* n*, then the probability that, if we obtain* $(\boldsymbol{u},\boldsymbol{e}) \leftarrow \mathsf{KG}(\boldsymbol{A})$*, then* $\boldsymbol{u} \in U$ *is negligible in* n*.*

We now prove Theorem 6.7.3.

Proof of Theorem 6.7.3 for $* = p$*.* We construct $\mathcal{A}$ solving $\mathrm{SIS}_{q,m,\sqrt{m}}$ on the average from an impersonator $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$ which succeeds impersonation under concurrent attack with non-negligible probability ϵ. Notice that the protocol is witness-indistinguishable since we set $m \geq 2(1+\delta)n \log q$ and $\mathsf{LNIC} = \mathit{Com}_{\boldsymbol{A}}$ is statistically-hiding and computationally-binding commitment scheme.

For the clarity, we write the transcript of interaction by $(\mathit{cmt}, \mathit{ch}, \mathit{rsp}, \mathit{dec})$. Since the protocol is parallelized, each cmt, ch, and rsp is an ordered list which contains t elements. For example, $\mathit{cmt} = (\mathit{cmt}_1, \ldots, \mathit{cmt}_t)$.

Given $\boldsymbol{A}$, $\mathcal{A}$ chooses a random secret key $\boldsymbol{e} \in \mathcal{S}(m,w)$ and computes $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e}$. Using the secret key, it can simulate the prover oracle perfectly. $\mathcal{A}$ runs CV on input $(\boldsymbol{A},\boldsymbol{u})$ and obtains a state for CP. $\mathcal{A}$ feeds the state to CP and acts as a legitimate verifier. Receiving commitments cmt, $\mathcal{A}$ chooses three challenges $\mathit{ch}^{(1)}$, $\mathit{ch}^{(2)}$, and $\mathit{ch}^{(3)}$ from $\{1,2,3\}^t$ uniformly at random. Rewinding with three challenges, $\mathcal{A}$ obtains three transcripts $(\mathit{cmt}, \mathit{ch}^{(i)}, \mathit{rsp}^{(i)}, \mathit{dec}^{(i)})$ for $i = 1,2,3$ as the results of the interactions.

By the Heavy Row Lemma [OO98], the probability that all $\mathit{dec}^{(i)}$ are 1 is at least $(\epsilon/2)^3$. Meanwhile, we have

$$\Pr\left[\exists j \in [t] : \{\mathit{ch}_j^{(1)}, \mathit{ch}_j^{(2)}, \mathit{ch}_j^{(3)}\} = \{1,2,3\}\right] = 1 - (7/9)^t$$

by a simple calculation, where $\mathit{ch}^{(i)}$ is randomly chosen from $\{1,2,3\}^t$. Thus the probability that $\mathcal{A}$ has three transcripts $(\mathit{cmt}, \mathit{ch}^{(i)}, \mathit{rsp}^{(i)}, \mathit{dec}^{(i)})$ for $i = 1,2,3$ such that $\mathit{dec}^{(i)} = 1$ for all i, and $\{\mathit{ch}_j^{(1)}, \mathit{ch}_j^{(2)}, \mathit{ch}_j^{(3)}\} = \{1,2,3\}$ for some $j \in [t]$ is at least $(\epsilon/2)^3 - (7/9)^t$, which is non-negligible since ϵ is non-negligible and $t = \omega(\log n)$.

We next show how $\mathcal{A}$ obtains a secret key or finds a collision of the hash functions in the string commitment scheme by using three good transcripts. Assume that $\mathcal{A}$ has three transcripts $(\mathit{cmt}^{(i)}, \mathit{ch}^{(i)}, \mathit{rsp}^{(i)}, \mathit{dec}^{(i)})$ for $i = 1,2,3$ such that $\mathit{cmt}^{(1)} = \mathit{cmt}^{(2)} = \mathit{cmt}^{(3)}$, $\mathit{dec}^{(i)} = 1$ for all i, and $\{\mathit{ch}_j^{(1)}, \mathit{ch}_j^{(2)}, \mathit{ch}_j^{(3)}\} = \{1,2,3\}$ for some $j \in [t]$. Without loss of generality, we assume that $\mathit{ch}_j^{(i)} = i$. We parse $\mathit{rsp}_j^{(i)}$ as in Step V2. We have following equations (We omit j for simplification):

$$\begin{array}{lll} c_1 & = \mathit{Com}_{\boldsymbol{A}}(\phi, \boldsymbol{A}\boldsymbol{y} - \boldsymbol{u}; \rho_1^{(2)}) & = \mathit{Com}_{\boldsymbol{A}}(\psi, \boldsymbol{A}\boldsymbol{z}; \rho_1^{(3)}), \\ c_2 & = \mathit{Com}_{\boldsymbol{A}}(\boldsymbol{x}; \rho_2^{(1)}) & = \mathit{Com}_{\boldsymbol{A}}(\psi(\boldsymbol{z}); \rho_2^{(3)}), \\ c_3 & = \mathit{Com}_{\boldsymbol{A}}(\boldsymbol{w} + \boldsymbol{x}; \rho_3^{(1)}) & = \mathit{Com}_{\boldsymbol{A}}(\phi(\boldsymbol{y}); \rho_3^{(2)}), \\ \boldsymbol{w} & \in \mathcal{S}(m,w). & \end{array}$$

If there exists a distinct pair of arguments of $\mathit{Com}_{\boldsymbol{A}}$, $\mathcal{A}$ violates the computational-binding property of LNIC and obtains a collision for $h_{\boldsymbol{A}}$ and, thus, solves $\mathrm{SIS}_{q,m,\sqrt{m}}$.

Next, we suppose that there exist no distinct pairs of the arguments of $Com_{\boldsymbol{A}}$. Let π denote the inverse permutation of ϕ. From the first equation, we have $\pi^{-1} = \phi = \psi$. Thus, we obtain $\boldsymbol{y} = \pi(\boldsymbol{w} + \boldsymbol{x})$ from the third equation. Combining it with the first equation, we have $\boldsymbol{Az} = \boldsymbol{A}(\pi(\boldsymbol{w}) + \pi(\boldsymbol{x})) - \boldsymbol{u}$. Since $\boldsymbol{z} = \phi^{-1}(\boldsymbol{x}) = \pi(\boldsymbol{x})$ from the second equation, we obtain $\boldsymbol{u} = \boldsymbol{A} \cdot \pi(\boldsymbol{w})$. Since $\boldsymbol{w} \in \mathcal{S}(m, w)$, so $\pi(\boldsymbol{w})$ also is in $\mathcal{S}(m, w)$. Therefore, $\mathcal{A}$ sets $\boldsymbol{e}' := \pi(\boldsymbol{w})$.

We now have to show that $\boldsymbol{e}' \neq \boldsymbol{e}$ with probability at least $1/2$. By Lemma 6.7.4, there must be another secret key $\boldsymbol{e}'$ corresponding to $\boldsymbol{u}$ with overwhelming probability. Recall that the protocol is statistically witness indistinguishable. Hence, $\mathcal{I}$'s view is independent of $\mathcal{A}$'s choice of $\boldsymbol{e}$ with overwhelming probability. Thus we have $\boldsymbol{e}' \neq \boldsymbol{e}$ with probability at least $1/2 - \mathsf{negl}(n)$. In this case $\mathcal{A}$ outputs $\boldsymbol{e} - \boldsymbol{e}'$ and solves $\mathrm{SIS}_{q,m,\sqrt{m}}$. □

We note that the above proof is extended into multi-user settings as in the proof of Lyubashevsky [Lyu08a].

We next show the proof for sequential composition. We will estimate the lower bound of the case where the adversary can answer the three challenge.

Proof of Theorem 6.7.3 for $* = s$. We note that the proof for the sequential composition is also very similar to the ones of Stern [Ste96] and Pointcheval and Poupard [PP03].

Assume that there exists a polynomial-time impersonator $\mathcal{I}$ that impersonates the prover with probability ϵ. We construct the polynomial-time algorithm $\mathcal{K}$ outputting three transcripts $(cmt, ch^{(i)}, rsp^{(i)}, dec^{(i)})$ such that $ch^{(i)} = i$ and $dec^{(i)} = 1$ for $i = 1, 2, 3$ with non-negligible probability. The algorithm yields an adversary $\mathcal{A}$ which violates the binding property of LNIC or the collision-resistance property of $h_{\boldsymbol{A}}$ as in the previous proof.

We describe the algorithm $\mathcal{K}$. On input $\boldsymbol{A}$, $\mathcal{K}$ chooses the random tape ω of the impersonator $\mathcal{I}$ and its own random tape for the learning phase. Using them, $\mathcal{K}$ terminate the setup and learning phases and obtains the state for CP. Next, it runs CP with several rewinds. Let I denote the random challenge of the legitimate verifier that is identified with the challenge $C = (ch_1, \dots, ch_t) \in \{0, 1, 2\}^t$. Consider the execution tree $T(\omega)$, corresponding to all accepted I, with a fixed ω. $\mathcal{K}$ finds a node of the tree which has three sons by (1) choose I uniformly at random, (2) check I contains a node with three sons by rewinding the prover (3) output three transcripts on the three sons. This yields $3t$ times of the executions of the basic protocol and thus $\mathcal{K}$ runs in polynomial time of n.

We next estimate the probability that $\mathcal{K}$ correctly outputs three valid transcripts. Let us denote by S the set of the pairs (ω, I) which lead to acceptance. Hence, we have that $\Pr_{(\omega,I)}[(\omega, I) \in S] = \epsilon = (2/3)^t + \epsilon'$. Next, we define the set $\Omega = \{\omega \mid \Pr_I[(\omega, I) \in S] \geq (2/3)^t + \epsilon'/2\}$. A standard argument shows that $\Pr_\omega[\omega \in \Omega] \geq \epsilon'/2$ and $\Pr[\Omega \mid S] \geq \epsilon'/2\epsilon$. Assume in the following that the event Ω occurs.

We denote by n_i the number of the nodes at the depth $i = 0, \dots, t$ of the tree $T(\omega)$. We know that $n_0 = 1$ and $n_t = 2^t + 3^t\epsilon'/2$, because $n_k/3^k = \Pr_I[(\omega, I) \in S] \geq$

$(2/3)^t + \epsilon'/2$. So, we have that

$$\prod_{i=0}^{t-1} \frac{n_{i+1}}{n_i} = \frac{n_t}{n_0} \geq 2^t + \frac{\epsilon'}{2} \cdot 3^t \geq \left(1 - \frac{\epsilon'}{2}\right) \cdot 2^t + \frac{\epsilon'}{2} \cdot 3^t.$$

By taking the logarithm of the inequation and using the convexity of the logarithm, we obtain that

$$\sum_{i=0}^{t-1} \log \frac{n_{i+1}}{n_i} \geq \left(1 - \frac{\epsilon'}{2}\right) \cdot \log 2^t + \frac{\epsilon'}{2} \cdot \log 3^t \geq t\left(\log 2 + \frac{\epsilon'}{2} \log \frac{3}{2}\right).$$

Therefore, there exists $i < t$ such that

$$\frac{n_{i+1}}{n_i} \geq 2(3/2)^{\epsilon'/2} = 2\exp\left(\frac{\epsilon'}{2} \cdot \log \frac{3}{2}\right) \geq 2 \cdot \left(1 + \frac{\epsilon'}{2} \cdot \log \frac{3}{2}\right) \geq 2 \cdot \left(1 + \frac{\epsilon'}{5}\right).$$

Let f_i and t_i denote the number of nodes at depth i with exactly 3 sons and that with at most 2 sons, respectively: We have that

$$n_i = f_i + t_i \text{ and } n_{i+1} \leq 3f_i + 2t_i = f_i + 2n_i.$$

Therefore, for the above i, we obtain that $2 + f_i/n_i \geq n_{i+1}/n_i 2 + 2\epsilon'/5$. Thus, so, with probability greater than $2\epsilon'/5$, the path I contains a node with 3 sons.

This shows that, with probability greater than $\epsilon(\epsilon'/2\epsilon)(2\epsilon'/5) = \epsilon'^2/5$, $\mathcal{K}$ finds a node with 3 sons.

□

6.7.3 The Cyclic/Ideal Version

We obtain the ID scheme $\mathsf{St\text{-}ID}^+_{\mathsf{C/IL},*}$ by combining the above setup and key-generation algorithms and the string commitment scheme with Stern's scheme as in Section 6.7. One can prove the securities of the schemes in the same manner to the proofs of Theorem 6.7.3. For simplicity, we only consider $\mathbf{f} = x^n + 1$ with $n = 2^k$.

Theorem 6.7.5. *Let m, q, and w be polynomially bounded functions of n such that $m > 4\log q$, q is a prime with $q \equiv 3 \pmod 8$, and $q^n / |\mathcal{S}(mn, w)|$ is negligible in n. Then, if $\mathbf{f}\text{-SIS}^\infty_{q,m,1}$ is hard on average, the ID scheme $\mathsf{St\text{-}ID}^+_{\mathsf{C/IL},*}$ is concurrently secure.*

In addition, let $m = m(n)$, $q = q(n)$, and $w = w(n)$ be polynomially bounded functions such that $q > 6mn^{3/2}\log n$, and $q^n / |\mathcal{S}(mn, w)|$ is negligible in n. Then for $\gamma = 72mn\log^2 n$, if $\mathbf{f}\text{-SVP}^\infty_\gamma$ is hard in the worst case then the ID scheme $\mathsf{St\text{-}ID}^+_{\mathsf{C/IL},*}$ is concurrently secure.

Proof Sketch. Notice that, by the hypothesis, ILNIC is statistically-hiding and computationally-binding under the assumption that $\mathbf{f}\text{-SIS}^\infty_{q,m,1}$ is hard on average.

As in the proofs of Theorem 6.7.3, we need to show that if there exists an impersonator $\mathcal{I}$ which succeeds impersonation under concurrent attack with non-negligible probability ϵ, there exists $\mathcal{A}$ that finds a collision $(\boldsymbol{e}_1, \boldsymbol{e}_2)$ for $\mathcal{H}_{\mathcal{I}}(\mathbf{f}, q, m)$ or violates the computational binding of ILNIC. The proofs are indeed the same as the proofs of Theorem 6.7.3 and we omit them. □

6.8 The Lyubashevsky ID Scheme – 2

The scheme is can be interpreted as a parallel composition of Ly08 with a hash family $\mathcal{H}(x^n + 1, q, m)$.

Recall the the Schnorr protocol [Sch91]. By extending the challenge set from $\{0, 1\}$ to $[0, C - 1]$, the soundness is reduced to $1/C$ rather than $1/2$. Lyubashevsky also extending the challenge set from $\{0, 1\}$ to $\{0, 1\}^n$, because the ideal-lattice-based hash functions is $R_{\mathbf{f},q}$-linear;

$$h_{\breve{a}}(\mathbf{c} \otimes \breve{e} + \breve{r}) = \sum_i \mathbf{a}_i \otimes (\mathbf{c} \otimes \mathbf{e}_i + \mathbf{r}_i) = \mathbf{c} \otimes \sum_i \mathbf{a}_i \otimes \mathbf{e}_i + \sum_i \mathbf{a}_i \otimes \mathbf{r}_i = \mathbf{c} \otimes h_{\breve{a}}(\breve{e}) + h_{\breve{a}}(\breve{r}).$$

See the protocol description.

6.8.1 Description

Let us fix $\mathbf{f} = x^n + 1$ in the following.

Scheme 6.8.1 (Ly09 [Lyu09]). All of the participants agree with the parameters $m = m(n)$, $q = q(n)$, $\sigma = \sigma(n)$, and $\kappa = \kappa(n)$ and the following sets D, D_e, D_r, D_c, and G;

- $D = \{\breve{g} \in R_{\mathbf{f},q}^m : \|\breve{g}\|_\infty \le mn\sigma\kappa\}$,
- $D_e = \{\breve{g} \in R_{\mathbf{f},q}^m : \|\breve{g}\|_\infty \le \sigma\}$,
- $D_r = \{\breve{g} \in R_{\mathbf{f},q}^m : \|\breve{g}\|_\infty \le mn\sigma\kappa\}$,
- $D_c = \{\breve{g} \in R_{\mathbf{f},q} : \|\breve{g}\|_1 \le \kappa\}$, and
- $G = \{\breve{g} \in R_{\mathbf{f},q}^m : \|\breve{g}\|_\infty \le mn\sigma\kappa - \sigma\kappa\}$.

Setup(1^n): The setup algorithm, given 1^n, outputs a random row vector $\breve{a} \leftarrow R_{\mathbf{f},q}^m$.

KeyGen($\breve{a}$): The key-generation algorithm, given the public parameter $\breve{a}$, chooses a random column vector $\breve{e} \in D_e$ and computes $\mathbf{u} \leftarrow h_{\breve{a}}(\breve{e}) \in R_{\mathbf{f},q}$. It outputs $(pk, sk) = (\mathbf{u}, \breve{e})$.

$\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$, $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$: The common inputs are $\breve{a}$ and $\mathbf{u}$. Prover's auxiliary input is $\breve{e}$. They interact as follows:

Step P1: Pick a random $\breve{r} \leftarrow D_r$ and send $\mathbf{y} \leftarrow h_{\breve{a}}(\breve{r})$.

Step V1: Send a random challenge $\mathbf{c} \leftarrow D_c$.

Step P2: Compute $\breve{z} \leftarrow \mathbf{c} \otimes \breve{e} + \breve{r}$. If $\breve{z} \in G^m$, then send it to the verifier. Otherwise, send $\perp$ and abort the protocol.

Step V2: Receiving $\check{z}$, accepts if $\check{z} \in G$ and $h_{\check{a}}(\check{z}) = \mathbf{u} \otimes \mathbf{c} + \mathbf{y}$.

It is obvious that, conditioned on that the prover does not abort, the honest prover is always accepted. He showed that the followings and proved the security by combining them.

1. The completeness error is at most $1 - 1/e$, that is, $\Pr_{\mathsf{P},\mathsf{V}}[dec = 1 : (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(\boldsymbol{A}, \boldsymbol{u}, \boldsymbol{e}) \leftrightarrow \mathsf{V}(\boldsymbol{A}, \boldsymbol{u})]] \geq 1/e$.
2. The protocol (P, V) is perfectly witness indistinguishable.
3. $\Pr_{\check{a} \leftarrow R^m_{\mathbf{f},q}, \check{e} \leftarrow D^m_e}[\exists\, \check{\boldsymbol{e}}' \in D^m_e, h_{\check{a}}(\check{\boldsymbol{e}}) = h_{\check{a}}(\check{\boldsymbol{e}}')] \geq 1 - \mathsf{negl}(n)$.
4. If we know $\check{\boldsymbol{e}}$ such that $h_{\check{a}}(\check{\boldsymbol{e}}) = \mathbf{u}$ and there is an adversary answering to two challenges $\mathbf{c}_1$ and $\mathbf{c}_2$ after committing $\mathbf{y}$, then we can retrieve a collision of $h_{\check{a}}$.

Theorem 6.8.2 ([Lyu09])**.** *Let* $\mathbf{f} = x^n + 1$. *The scheme* Ly09-ID *is concurrently secure if* $\mathbf{f}\text{-SIS}^{\infty}_{q,m,\beta}$ *with* $\beta = 2(mn - 1)\sigma\kappa$ *is hard on average. In particular, let* σ *be a constant and let* $\kappa(n) = \Theta(\log^2 n)$. *Then the scheme is secure if* $\mathbf{f}\text{-SVP}^{\infty}_{\gamma}$ *is hard in the worst case, where* $\gamma = \tilde{O}(n^2)$.

The first and third conditions are satisfied by careful choices of the parameters. The second part is complex and see [Lyu09]. The fourth part is almost obvious. Let $(\mathbf{y}, \mathbf{c}_1, \check{z}_1)$ and $(\mathbf{y}, \mathbf{c}_2, \check{z}_2)$ be two transcripts which lead acceptance such that $\mathbf{c}_1 \neq \mathbf{c}_2$. Then, we have $h_{\check{a}}(\check{z}_i) = \mathbf{c}_i \otimes h_{\check{a}}(\check{\boldsymbol{e}}) + \mathbf{y}$ and $h_{\check{a}}(\check{z}_1 - \mathbf{c}_1 \otimes \check{\boldsymbol{e}}) = h_{\check{a}}(\check{z}_2 - \mathbf{c}_2 \otimes \check{\boldsymbol{e}})$. Thus, $(\check{z}_1 - \mathbf{c}_1 \otimes \check{\boldsymbol{e}}, \check{z}_2 - \mathbf{c}_2 \otimes \check{\boldsymbol{e}})$ seems a collision for $h_{\check{a}}$ and both are in D. (We need to show that they differs but we omit it.) For the details of the parameters and the proofs, see the original paper [Lyu09].

The ID scheme has completeness error $1 - 1/e$ if we carefully choose the parameters. Hence, this protocol should be composed in parallel to reduce the completeness error to $\mathsf{negl}(n)$. In order to decrease the communication cost, one can use the hash-based commitment $H(\cdot)$, where H is any collision-resistant hash function.

6.9 Summary

We have reviewed several ID scheme based on lattice and ideal lattice problems and their security. As a summary, see Table 6.1.

The MV protocol: The variants based on the MV protocol requires the mild assumption, $\mathrm{SIVP}_{\tilde{O}(n^{1.5})}$ is hard in the worst case. In addition, the one of variants directly bears an identity-based identification scheme. See Chapter 7.

The KTX ID scheme: The assumption is the weakest among those in other schemes. The one weak point is a long transcript. The scheme requires a permutation over $[m]$ and thus, the communication cost is most expensive.

Lattice-based ID schemes ($\boldsymbol{A}_0, \boldsymbol{A}_1, \boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$)						
	Par.	Public key	Relation	γ in GapSVP$_\gamma$	Comm. cost	Errors
$\mathsf{MV\text{-}ID}^{+}_{\mathsf{GL,p}}$ [MV03]	–	$\boldsymbol{A}_0, \boldsymbol{A}_1$	$\boldsymbol{A}_0\boldsymbol{e} = \mathbf{0}$ or $\boldsymbol{A}_1\boldsymbol{e} = \mathbf{0}$	$\tilde{O}(n^{1.5})$	$t \cdot \tilde{O}(n)$	1-sided
$\mathsf{MV\text{-}ID}^{++}_{\mathsf{GL,p}}$	$\boldsymbol{A}$	$\boldsymbol{u}$	$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u}$	$\tilde{O}(n^{1.5})$	$t \cdot \tilde{O}(n)$	1-sided
$\mathsf{Ly08\text{-}ID}_{\mathsf{GL,p}}$ [Lyu08a]	$(\boldsymbol{A})$	$\boldsymbol{A}, \boldsymbol{u}$	$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u}$	$\tilde{O}(n^2)$	$t \cdot \tilde{O}(n)$	2-sided
$\mathsf{St\text{-}ID}_{\mathsf{GL,p}}$ [KTX08]	$\boldsymbol{A}$	$\boldsymbol{u}$	$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u}$ and $w_H(\boldsymbol{e}) = w$	$\tilde{O}(n)$	$t \cdot \tilde{O}(n)$	1-sided

Ideal-Lattice-based ID schemes ($\mathbf{f} = x^n + 1$ and $\check{\boldsymbol{a}}_0, \check{\boldsymbol{a}}_1, \check{\boldsymbol{a}} \in R_{\mathbf{f},q}^m$)						
	Par.	Public key	Relation	γ in $\mathbf{f}$-SVP$_\gamma^\infty$	Comm. cost	Errors
$\mathsf{MV\text{-}ID}^{+}_{\mathsf{C/IL,p}}$ [MV03]	–	$\check{\boldsymbol{a}}_0, \check{\boldsymbol{a}}_1$	$\check{\boldsymbol{a}}_0\check{\boldsymbol{e}} = 0$ or $\check{\boldsymbol{a}}_1\check{\boldsymbol{e}} = 0$	$\tilde{O}(n^{1.5})$	$t \cdot \tilde{O}(n)$	1-sided
$\mathsf{MV\text{-}ID}^{++}_{\mathsf{C/IL,p}}$	$\check{\boldsymbol{a}}$	$\mathbf{u}$	$\check{\boldsymbol{a}}\check{\boldsymbol{e}} = \mathbf{u}$	$\tilde{O}(n^{1.5})$	$t \cdot \tilde{O}(n)$	1-sided
$\mathsf{Ly08\text{-}ID}_{\mathsf{C/IL,p}}$ [Lyu08a]	$(\check{\boldsymbol{a}})$	$\check{\boldsymbol{a}}, \mathbf{u}$	$\check{\boldsymbol{a}}\check{\boldsymbol{e}} = \mathbf{u}$	$\tilde{O}(n^2)$	$t \cdot \tilde{O}(n)$	2-sided
$\mathsf{St\text{-}ID}_{\mathsf{C/IL,p}}$ [KTX08]	$\check{\boldsymbol{a}}$	$\mathbf{u}$	$\check{\boldsymbol{a}}\check{\boldsymbol{e}} = \mathbf{u}$ and $w_H(\check{\boldsymbol{e}}) = w$	$\tilde{O}(n)$	$t \cdot \tilde{O}(n)$	1-sided
$\mathsf{Ly09\text{-}ID}_{\mathsf{p}}$ [Lyu09]	$(\check{\boldsymbol{a}})$	$\check{\boldsymbol{a}}, \mathbf{u}$	$\check{\boldsymbol{a}}\check{\boldsymbol{e}} = \mathbf{u}$	$\tilde{O}(n^2)$	$t \cdot \tilde{O}(n)$	2-sided

Table 6.1: Comparisons among ID schemes. A secret key *sk* is $\boldsymbol{e} \in D_n$. The factor n denotes the security parameter. Assume that the protocols are repeated t times in parallel for reducing errors. In $\mathsf{MV\text{-}ID}$, we set $\delta = \sqrt{m}$ and $\delta = \sqrt{mn}$ with respect to $\mathsf{L} = \mathsf{GL}$ and $\mathsf{C/IL}$, respectively.

The Lyubashevsky ID schemes: The assumption is strongest one. However, it attracts us by its low communication cost. In addition, the variant $\mathsf{Ly09\text{-}ID}$ yields a simple signature scheme by applying the Fiat–Shamir transform. See Chapter 11.

7
Identity-based Identification

In this chapter, we show that the combination of the Micciancio-Vadhan identification with lattice-based signatures in Chapter 11 yields concurrently secure identity-based identification scheme in the random oracle model.

Organization: In Section 7.1, we review a model of identity-based identification (IBI) schemes and security notions of them. In Section 7.2, we construct IBIs and prove their security.

7.1 Definitions

7.1.1 Model of Identity-Based Identification Schemes

Identity-based cryptosystems (precisely, encryption and signature schemes) are proposed by Shamir [Sha85]. First, a master generates the public parameters and corresponding master's secret key. Each user has no public key but identity. They use an identity instead of public key in the cryptosystem. Notice that anyone of user obtains its secret key from the master, called as user's secret key.

On identity-based identification, the prover has a user secret key as its auxiliary input and the verifier is given the public parameter and prover's identity as input.

We adopt the definition by Bellare, Namprempre, and Neven [BNN09]. Formally, an identity-based identification scheme IBI is a quadruplet of algorithms $(\mathsf{Setup}, \mathsf{Ext}, \mathsf{P}, \mathsf{V})$.

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs public parameters *param* and a master secret key *msk*.

$\mathsf{Ext}(msk, id)$**:** An extraction algorithm, given msk and an identity id, outputs a secret key sk_{id} for the identity id.

$\mathsf{P}(param, id, sk_{id})$**,** $\mathsf{V}(param, id)$**:** (P, V) is an interactive protocol. A prover algorithm P takes $param$, id, and sk_{id} as inputs. A verifier algorithm V takes $param$ and id as inputs. At the end of interaction, V outputs 0 (reject) or 1 (accept).

We require the natural correctness condition; For any $param$ generated by $\mathsf{Setup}(1^n)$ and sk_{id} generated by $\mathsf{Ext}(msk, id)$, the decision of $\mathsf{V}(param, id)$ interacting with $\mathsf{P}(param, id, sk_{id})$ is 1 with probability 1. That is, for any id

$$\Pr\left[dec = 1 : \begin{array}{l} (param, msk) \leftarrow \mathsf{Setup}(1^n); \\ sk_{id} \leftarrow \mathsf{Ext}(msk, id); \\ (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(param, id, sk_{id}) \leftrightarrow \mathsf{V}(param, id)]; \end{array} \right] = 1.$$

An IBI scheme IBI is said to be *canonical* if the protocol is 3-move and public coin as in Section 6.2.

7.1.2 Security Notions

The definition of security notions are almost identical to these of ID schemes. We describe the formal definition as follows. Consider the experiment $\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{IBI},\mathcal{A}}(n)$ between the challenger C and the impersonator $\mathcal{A} = (\mathsf{CV}, \mathsf{CP})$, where $\text{atk} \in \{\text{pa}, \text{aa}, \text{ca}\}$.

Experiment $\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{IBI},\mathcal{A}}(n)$**:**

Setup Phase: The challenger C obtains $(param, msk) \leftarrow \mathsf{Setup}(1^n)$. Next, C sets $HU, CU, TU \leftarrow \emptyset$ and $PS \leftarrow \emptyset$, where HU, CU, TU and PS denotes the set of honest users, corrupted users, target users, and provers' sessions, respectively. The impersonator CV is given the security parameter 1^n, the system parameter $param$.

Learning Phase: The impersonator CV can query to the oracles Init, Extract, and Prov.

- The oracle Init receives an identity id. If $id \in HU \cup CU \cup TU$ then return $\perp$. Otherwise, it obtains $sk_{id} \leftarrow \mathsf{Ext}(msk, id)$, stores it into $usk[id]$, and adds id to HU. Finally, return 1 to the adversary.
- The oracle Extract receives an identity id. If $id \notin HU$ then return $\perp$. Else, it adds id to CU, deletes id from HU, and returns $usk[id]$ to the adversary.
- The oracle Prov receives inputs id, s, M_{in}, where s denotes the session identifier. This oracle changes its behavior in three attacks.
  - If $\text{atk} = \text{pa}$, it obtains $(tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(param, id, sk_{id}) \leftrightarrow \mathsf{V}(param, id)]$ and returns (tr, dec) to the adversary.

- If atk $=$ aa, it runs as follows: If $id \notin HU$ then return $\perp$. If $(id, s) \notin PS$ then it sets $PS \leftarrow \{(id, s)\}$, picks a random coin ρ, and sets a state of the prover $st_{\mathsf{P}}[s] \leftarrow (param, sk_{id}, \rho)$. Next, it obtains $(M_{out}, st_{\mathsf{P}}[id, s]) \leftarrow \mathsf{P}(M_{in}, st_{\mathsf{P}}[id, s])$. It returns M_{out}.
- If atk $=$ ca, it runs as follows: If $(id, s) \notin PS$ then it adds (id, s) to PS (that is, $PS \leftarrow PS \cup \{(id, s)\}$), picks a random coin ρ, and sets a state of the prover $st_{\mathsf{P}}[id, s] \leftarrow (param, sk_{id}, \rho)$. Next, it obtains $(M_{out}, st_{\mathsf{P}}[id, s]) \leftarrow \mathsf{P}(M_{in}, st_{\mathsf{P}}[id, s])$. It returns M_{out}.

At the end of the phase CV outputs (id^*, st_{CP}).

Challenge Phase: Suppose that C receives (id^*, st_{CP}) from CV. If $id^* \notin HU$, then C outputs 0 and halts. Else, the challenger C adds id^* to TU, deletes id^* from HU, and gives st_{CP} to CP. Finally, the challenger obtains $(tr, dec) \leftarrow \mathbf{Run}[\mathsf{CP}(st_{\mathsf{CP}})^{\text{Init,Extract,Prov}} \leftrightarrow \mathsf{V}(param, id)]$ and returns dec.

Notice that if atk $=$ pa the adversary could learn only transcripts between the legitimate prover and verifier. If atk $=$ aa, the adversary could interact with the legitimate prover sequentially and has the power to abort the session. If atk $=$ ca, the adversary interact with the legitimate prover concurrently by indicating each interaction with session identifier.

Definition 7.1.1. Let $\mathsf{IBI} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{P}, \mathsf{V})$ be an IBI scheme, $\mathcal{A} = (\mathsf{CV}, \mathsf{CP})$ an impersonator, and n a security parameter. We define the advantage of $\mathcal{A}$ as $\mathbf{Adv}^{\text{imp-atk}}_{\mathsf{IBI},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\text{imp-atk}}_{\mathsf{IBI},\mathcal{A}}(n) = 1\right]$. We say that IBI is secure against impersonation under passive, active, and concurrent attacks if $\mathbf{Adv}^{\text{imp-atk}}_{\mathsf{IBI},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$ where atk $=$ pa, aa, ca, respectively.

7.2 Identity-based Identification Schemes

An intuitive explanation of a well-known strategy to construct an IBI scheme is as follows: A master generates (vk, sk) which is a verification and a signing keys of a signature scheme. It publishes vk and keeps sk secret. If a user queries with id, then the master returns a signature σ on id. As identification, the user proves possession of σ using some protocol. We note that Bellare, Namprempre, and Neven [BNN09] also gave the general construction of IBIs from any identification scheme and any signature scheme. For more information on generic constructions, see [KH04, BNN09, YCW$^+$07].

They are several identity-based identification schemes based on number theoretic problems. However, we know a few IBI schemes based on the combinatorial problems with security proofs. The one is that by Cayrel, Gaborit, Galindo, and Girault [CGDG09] (see also [CGG07]) based on the coding problems, which consists of the Courtois–Finiasz–Sendrier signature scheme [CFS01] and Stern's ID scheme (Section 6.6). The other is one by Stehlé, Steinfeld, Tanaka, and Xagawa [SSTX09], which is obtained by combining the Gentry–Peikert–Vaikuntanathan (GPV) signature scheme [GPV08] (Section 11.3) and the Micciancio–Vadhan (MV) protocol (Section 6.3).

We now give a brief review of the GPV signature scheme (For the details, Chapter 10 and Section 11.3). Roughly speaking, the public key is $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and the secret key is a short basis $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ of a lattice $\Lambda = \Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\tilde{\boldsymbol{T}}\| \le L$. If such basis is known, one can sample $D_{\Lambda,s,\boldsymbol{t}}$ with $s = L \cdot \omega(\sqrt{\log n})$ and for any $\boldsymbol{t} \in \mathbb{R}^m$ (see Chapter 10). This indicates the trapdoor $\boldsymbol{T}$ allows us to sample $\boldsymbol{e} \leftarrow D_{\Lambda,s}$ conditioned on $\boldsymbol{u} = h_{\boldsymbol{A}}(\boldsymbol{e})$. Notice that the sample $\boldsymbol{e}$ has a norm at most $s\sqrt{m}$ with overwhelming probability.

We follow the above general strategy to construct lattice-based IBIs: Master's key pair is $(vk = \boldsymbol{A}, sk = \boldsymbol{T})$. A signature on id is $\sigma = \boldsymbol{e}$ such that $\boldsymbol{Ae} \equiv H(id) \pmod q$ and $\boldsymbol{e} \in D_n = \{\boldsymbol{e} \in \mathbb{Z}^m \mid \|\boldsymbol{e}\| \le s\sqrt{m}\}$. We then use the MV protocol for a proof of signature possession. Since the signature schemes are secure under the worst-case hardness of lattice problems and the MV protocol is witness indistinguishable and proof-of-knowledge, the IBI schemes also enjoy the concurrent security in the random oracle model.

Scheme 7.2.1 (LIBI [SSTX09]). See GPV-FDH = (Setup, KeyGen, Sign, Ver) in Section 11.3. Let (P, V) be the prover and the verifier in the scheme $\mathsf{MV}^{+}_{\mathsf{GL}}$. Let $H : \{0,1\}^* \to \mathbb{Z}_q^n$ be the random oracle which is used in GPV-FDH. The IBI scheme LIBI = (Setup′, Ext′, P′, V′) is defined as follows:

Setup′(1^n)**:** Given the security parameter 1^n, the setup algorithm obtains $(\boldsymbol{A}, \boldsymbol{T}) \leftarrow \mathsf{KeyGen}(1^n)$, where $\boldsymbol{A}$ is almost uniformly distributed over $\mathbb{Z}_q^{n\times m}$ and $\boldsymbol{T}$ is a short basis of $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\tilde{\boldsymbol{T}}\| \le L$ for some L. It outputs $(param, msk) = (\boldsymbol{A}, (\boldsymbol{A}, \boldsymbol{T}))$.

Ext′$((\boldsymbol{A}, \boldsymbol{T}), id)$**:** Given an identity id, it outputs $\boldsymbol{e} \leftarrow \mathsf{Sign}((\boldsymbol{A}, \boldsymbol{T}), id)$ such that $\|\boldsymbol{e}\|$ is short ($\|\boldsymbol{e}\| \le d_2 = s\sqrt{m}$) and $\boldsymbol{Ae} \equiv H(id) \bmod q$.

P′ **and** V′**:** The common inputs are $\boldsymbol{A}$ and id. Prover's auxiliary input is $\boldsymbol{e}$. Let us define $\boldsymbol{u} = H(id) \in \mathbb{Z}_q^n$. They are the same as P and V with parameter d_2 in the MV protocol.

Theorem 7.2.2. *The obtained IBI scheme is concurrently secure in the random oracle model if* $\mathrm{SIS}^2_{q,m,O(\sqrt{m}\cdot d_2)}$ *is hard on the average, where* $d_2 = L\sqrt{m}\cdot\omega(\sqrt{\log n})$.

The GPV signature scheme [GPV08] with the Alwen–Peikert construction in Chapter 11 yields that $L = O(\sqrt{n \log q})$ when $m = (5+3\delta)n\log q$ for some constant $\delta > 0$. Thus, the security of the IBI scheme is reduced to SIVP_γ for $\gamma = \tilde{O}(n^2)$.

In addition, if we replace the GPV signature scheme with the Bonsai signature schemes in Section 11.7, the IBI schemes (and hierarchy IBI scheme) in the standard model are obtained. If we employ the ideal-lattice-based signature schemes, we also obtain the ideal-lattice-based IBI (and hierarchy IBI) schemes which are concurrently secure.

8 Ring Identification

Organization: Section 8.1 introduces rind identification. Section 8.2 defines a model and security notions of rind identification schemes. Section 8.3 gives details of the Kawachi–Tanaka–Xagawa ring identification schemes.

8.1 Introduction

Dodis, Kiayias, Nicolosi, and Shoup introduced new identification, under the name ad hoc anonymous identification (AID) [DKNS04], which are identification versions of ring signature[1]. We call this new identification "ring identification scheme" rather than "ad hoc anonymous identification scheme" for simplification of the name and stress of the relation to ring signature.

An RID scheme allows a user to anonymously prove his/her membership in a ring, which is the set of public keys, if and only if the user is an actual member of the ring. We use the term "ring" instead of "group," since we want to stress that the ring is formed in an ad hoc fashion, without help of the group manager. Hence, we then assume that every user registers his/her public key to the public key infrastructure.

RID schemes: By taking OR of l statements [DSDCPY94], we can straightforwardly obtain an $\mathsf{MV}^{+}_{\mathsf{GL}}$-based RID scheme,whose security is based on the worst-case hardness of lattice problem. The prover and the verifier have the common input $pk_1, \ldots, pk_l$. The prover convinces the verifier that he/she has a secret key corresponding to one of public keys, pk_i.

[1]Indeed, applying the Fiat-Shamir transform to their AID schemes, we can obtain ring signature schemes. See the original paper [DKNS04]

However, this simple modification requires a large overhead cost involving the size of the ring. Let l be the number of the members in the ring and n the security parameter. The protocol is run in t times in parallel to reduce the errors. The communication costs of the $\mathsf{MV}^+_{\mathsf{GL}}$-based scheme is $tl \cdot \tilde{O}(n)$. The size of the ring is $l \cdot \tilde{O}(n^2)$ in the modified versions of $\mathsf{MV}^+_{\mathsf{GL}}$.

On RID schemes, the KTX ring identification schemes $\mathsf{KTX\text{-}RID}_{\mathsf{GL}}$ by Kawachi et al. [KTX08] require many *vectors* proportional to the member of the ring, while the $\mathsf{MV}^+_{\mathsf{GL}}$-based scheme requires many *matrices* proportional to the size of the group (see Table 6.1). Additionally, the communication cost of $\mathsf{KTX\text{-}RID}_{\mathsf{GL}}$ is $t \cdot \tilde{O}(n + l)$, while those in the $\mathsf{MV}^+_{\mathsf{GL}}$-based is $tl \cdot \tilde{O}(n)$.

8.2 Definitions

8.2.1 Model of Ring Identification Schemes

An RID scheme is a sextuplet of algorithms $\mathsf{RID} = (\mathsf{Setup}, \mathsf{Reg}, \mathsf{RPKC}, \mathsf{RSKC}, \mathsf{P}, \mathsf{V})$:

$\mathsf{Setup}(1^n)$: A setup algorithm, given the security parameter 1^n, outputs public parameters *param*.

$\mathsf{Reg}(param, i)$: A key-generation algorithm, given *param* and user identity i, outputs a pair of a public key and a secret key (pk_i, sk_i). This models the key registration procedure.

$\mathsf{RPKC}(param, R = (pk_{i_1}, \ldots, pk_{i_l}))$: A ring public-key construction algorithm, given the public parameters *param*, a ring of public keys $R = (pk_{i_1}, \ldots, pk_{i_l})$, outputs a ring public key *rpk*.

$\mathsf{RSKC}(param, R = (pk_{i_1}, \ldots, pk_{i_l}), sk_{i_k})$: A ring secret-key construction algorithm, given *param*, $R = (pk_{i_1}, \ldots, pk_{i_l})$, and one of corresponding secret key sk_{i_k}, outputs a ring secret key *rsk*.

$\mathsf{P}(param, rpk, rsk)$, $\mathsf{V}(param, rpk)$: (P, V) is an interactive protocol. A prover algorithm P takes *param*, *rpk*, and *rsk* as inputs. A verifier algorithm V takes *param* and *rpk* as inputs. At the end of interaction, V outputs 0 (reject) or 1 (accept).

Correctness: As in the definition of ID schemes, we require the natural correctness condition; For any *param*, $\{(pk_i, sk_i)\}_{i=1,\ldots,l}$, *rpk*, and *rsk* generated by $\mathsf{Setup}(1^n)$, $\mathsf{Reg}(param)$, $\mathsf{RPKC}(\{pk_i\}_{i=1,\ldots,l}, sk_k)$, and $\mathsf{RSKC}(\{pk_i\}_{i=1,\ldots,l}, sk_k)$, the decision of $\mathsf{V}(param, rpk)$ interacting with $\mathsf{P}(param, rpk, rsk)$ is 1 with probability 1. That is, for any polynomial $Q = Q(n)$ and $\{i_1, \ldots, i_l\} \subseteq \{1, 2, \ldots, Q\}$, and

$k \in \{1, \ldots, l\}$,

$$\Pr\left[dec = 1 : \begin{array}{l} param \leftarrow \mathsf{Setup}(1^n); \\ (pk_i, sk_i) \leftarrow \mathsf{Reg}(param, i) \text{ for } i = 1, \ldots, Q; \\ rpk \leftarrow \mathsf{RPKC}(param, \{pk_i\}_{i=i_1,\ldots,i_l}); \\ rsk \leftarrow \mathsf{RSKC}(param, \{pk_i\}_{i=i_1,\ldots,i_l}, sk_{i_k}); \\ (tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(param, rpk, rsk) \leftrightarrow \mathsf{V}(param, rpk)]; \end{array} \right] = 1.$$

8.2.2 Security Notions

There are two goals for security of RID schemes: Security against impersonation and anonymity.

Dodis et al. formally defined security against impersonation under passive attack. They mentioned the definition of security against impersonation under concurrent attack. However, they did not give the formal definition (see [DKNS04, Section 3.2]). Thus, we define the security notion with respect to concurrent attack. In the setting of chosen-group attack, the adversary could force the prover to prove the membership in an arbitrary group if the prover is indeed a member of the group. Additionally, concurrent attack allows the cheating verifier to interact with the clones of any provers. Also, they allow the cheating prover to interact with the clones of provers, but prohibit it from interacting with the target provers. We say RID is secure against impersonation under concurrent chosen-group attack, if any polynomial-time adversary cannot impersonate the legitimate prover in the above settings.

The security notion, anonymity against full key exposure, captures the property that an adversary cannot distinguish two transcripts even if the adversary has the secret keys of all the members. We say RID is anonymous against full key exposure if any polynomial-time adversary cannot distinguish two provers with a common set of public keys even though the adversary generates all keys of the set.

Security against impersonation: In the setting of chosen-group attack, the adversary could force the prover to prove the membership in an arbitrary ring if the prover is indeed a member of the ring. Additionally, concurrent attack allows the cheating verifier to interact with the clones of any provers. Also, they allow the cheating prover to interact with the clones of provers, but prohibit it from interacting with anyone of the target provers. Notice that this condition prevents the adversary a simple Man-in-the-Middle attack.

We describe the formal definition of the security as follows. Consider the following experiment $\mathbf{Exp}^{\text{imp-cg-atk}}_{\mathsf{RID},\mathcal{I}}(n)$ between the challenger and the impersonator $\mathcal{I} = (\mathsf{CV}, \mathsf{CP})$, where $\text{atk} \in \{\text{pa}, \text{aa}, \text{ca}\}$.

Experiment $\mathbf{Exp}^{\text{imp-cg-atk}}_{\mathsf{RID},\mathcal{A}}(n)$:

Setup Phase: The challenger obtains $param \leftarrow \mathsf{Setup}(1^n)$ and initializes $HU, CU, TU, PS \leftarrow \emptyset$, where HU, CU, and TU denote the sets of honest users, corrupted users, and target users, respectively, and PS denotes the set of prover's sessions. The impersonator CV is given the security parameter 1^n and the system parameter $param$.

Learning Phase: The impersonator CV can query to the three oracles Init, Corr, and Prov.

- The oracle Init receives input i. If $i \in HU \cup CU \cup TU$ then returns $\perp$. Otherwise, it obtains $(pk_i, sk_i) \leftarrow \mathsf{Reg}(param, i; r_i)$, adds i to HU, and provides $\mathcal{A}$ with pk_i.
- The oracle Corr receives input i. If $i \notin HU \setminus TU$ then returns $\perp$. Otherwise, it adds i to CU, deletes i in HU, and returns r_i to $\mathcal{A}$.
- The oracle Prov receives some inputs. This oracle changes its behavior in three attacks.
  - If atk = pa, it receives inputs $R = (pk_{i_1}, \dots, pk_{i_l})$ and i_k. If $pk_{i_k} \notin R$ then returns $\perp$. (The public keys in R need not to be registered.) If the check is passed, it obtains $rpk \leftarrow \mathsf{RPKC}(param, R)$ and $rsk \leftarrow \mathsf{RSKC}(param, R, sk_{i_k})$, and $(tr, dec) \leftarrow \mathbf{Run}[\mathsf{P}(param, rpk, rsk) \leftrightarrow \mathsf{V}(param, rpk)]$. It returns (tr, dec) to the adversary.
  - If atk = aa, it receives inputs $R = (pk_{i_1}, \dots, pk_{i_l})$, i_k, s, and M_{in}. If $pk_{i_k} \notin R$ or $i_k \notin HU \setminus TU$ then returns $\perp$. (The public keys in R need not to be registered.) If the checks are passed, it obtains $rpk \leftarrow \mathsf{RPKC}(param, R)$ and $rsk \leftarrow \mathsf{RSKC}(param, R, sk_{i_k})$. If $(R, i_k, s) \notin PS$ then it sets $PS \leftarrow \{(R, i_k, s)\}$ to PS, picks a random coin ρ, and sets a state of the prover $st_{\mathsf{P}}[(R, i_k, s)] \leftarrow (param, R, rpk, rsk, sk_{i_k}, \rho)$. Next, it obtains $(M_{out}, st_{\mathsf{P}}[(R, i_k, s)]) \leftarrow \mathsf{P}(M_{in}, st_{\mathsf{P}}[(R, i_k, s)])$. Finally, it returns M_{out}.
  - If atk = ca, it receives inputs $R = (pk_{i_1}, \dots, pk_{i_l})$, i, s, and M_{in}. If $pk_{i_k} \notin R$ or $i_k \notin HU \setminus TU$ then returns $\perp$. (The public keys in R need not to be registered.) If the checks are passed, it obtains $rpk \leftarrow \mathsf{RPKC}(param, R)$ and $rsk \leftarrow \mathsf{RSKC}(param, R, sk_{i_k})$. If $(R, i_k, s) \notin PS$ then it adds (R, i_k, s) to PS ($PS \leftarrow PS \cup \{(R, i_k, s)\}$), picks a random coin ρ, and sets a state of the prover $st_{\mathsf{P}}[(R, i_k, s)] \leftarrow (param, R, rpk, rsk, sk_{i_k}, \rho)$. Next, it obtains $(M_{out}, st_{\mathsf{P}}[(R, i_k, s)]) \leftarrow \mathsf{P}(M_{in}, st_{\mathsf{P}}[(R, i_k, s)])$. Finally, it returns M_{out}.

Challenge Phase: CV outputs a set of public keys $R_t = (pk_{i_1}, \dots, pk_{i_l})$ and st_{CP}. If the indexes of the keys $\{i_1, \dots, i_l\} \nsubseteq HU$ then the challenger outputs 0 and halts. Otherwise, the challenger sets $TU \leftarrow \{i_1, \dots, i_l\}$ and gives st_{CP} to CP. CP can query to the oracles Init, Corr, and Prov as in the learning phase. Finally, the challenger obtains $(tr, dec) \leftarrow$

$\textbf{Run}[\mathsf{CP}(st_{\mathsf{CP}})^{\textsc{Init},\textsc{Corr},\textsc{Prov}} \leftrightarrow \mathsf{V}(param, R_t)]$ and outputs *dec*.

Definition 8.2.1. Let RID be an RID scheme and $\mathcal{A} = (\mathsf{CV}, \mathsf{CP})$ an impersonator. Let n be a security parameter. The advantage of $\mathcal{A}$ in attacking RID is defined by

$$\mathbf{Adv}^{\text{imp-cg-atk}}_{\mathsf{RID},\mathcal{A}}(n) = \Pr\left[\mathbf{Exp}^{\text{imp-cg-atk}}_{\mathsf{RID},\mathcal{A}}(n) = 1\right].$$

We say that RID is secure against impersonation under passive/active/concurrent chosen-group attack if $\mathbf{Adv}^{\text{imp-cg-atk}}_{\mathsf{RID},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$, where atk $=$ pa, aa, ca, respectively.

Anonymity against full key exposure: Anonymity against full key exposure for an RID scheme RID is defined by using the following experiment $\mathbf{Exp}^{\text{anon-fke}}_{\mathsf{RID},\mathcal{A}}(n)$ between a challenger and adversary $\mathcal{A}$:

Experiment $\mathbf{Exp}^{\text{anon-fke}}_{\mathsf{RID},\mathcal{A}}(n)$:

Setup Phase: The challenger runs the algorithm Setup with input 1^n and obtains *param*. The adversary $\mathcal{A}$ is given the system parameter *param*.

Challenge Phase: $\mathcal{A}$ requests a challenge by sending to the challenger the values $((pk_{i_0}, sk_{i_0}), (pk_{i_1}, sk_{i_1}), R)$. Here the set of public keys R contains pk_{i_0} and pk_{i_1}, and (pk_{i_0}, sk_{i_0}) and (pk_{i_1}, sk_{i_1}) are valid key pairs. The challenger chooses a random bit $b \in \{0, 1\}$ and runs the protocol as a prover who has sk_{i_b}. $(tr, b^*) \leftarrow \textbf{Run}[\mathsf{P}(param, R, sk_{i_b}) \leftrightarrow \mathcal{A}]$. If $b = b^*$ the challenger returns 1, otherwise returns 0.

Definition 8.2.2. Let RID be an RID scheme, $\mathcal{A}$ an adversary, and n a security parameter. The advantage of $\mathcal{A}$ in attacking RID is defined by

$$\mathbf{Adv}^{\text{anon-fke}}_{\mathsf{RID},\mathcal{A}}(n) := \left|\Pr\left[\mathbf{Exp}^{\text{anon-fke}}_{\mathsf{RID},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say that RID has anonymity with full key exposure if $\mathbf{Adv}^{\text{anon-fke}}_{\mathsf{RID},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time $\mathcal{A}$.

8.3 The Kawachi–Tanaka–Xagawa Ring Identification Schemes

We review the Kawachi–Tanaka–Xagawa RID scheme based on GapSVP. First, we sketch a basic idea for our construction: Let $\boldsymbol{A}$ be a system parameter. Each user has a secret key $\boldsymbol{e}_i \in \mathcal{S}(m, w)$ and a public key $\boldsymbol{u}_i = \boldsymbol{A}\boldsymbol{e}_i \bmod q$. In the RID scheme, a group is specified by a set of public keys $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l)$ of the members. Let $\boldsymbol{i}_{i,l}$ denote an l-dimensional vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ whose i-th element is 1. The prover in the group, who has a secret key $\boldsymbol{e}_i$, wants convinces the verifier that he/she knows that $\boldsymbol{e}' := \boldsymbol{e}_i \circ -\boldsymbol{i}_{i,l}$ such that $[\boldsymbol{A}\, \boldsymbol{u}_1 \ldots \boldsymbol{u}_l]\boldsymbol{e}' = \boldsymbol{0}$ and $\boldsymbol{e}_i \in \mathcal{S}(m, m/2)$. Changing the parameters and using Stern's protocol, the prover can convinces the verifier that he/she has $\boldsymbol{e}'$ such that $[\boldsymbol{A}\, \boldsymbol{u}_1\, \ldots\, \boldsymbol{u}_l]\boldsymbol{e}' = \boldsymbol{0}$, the numbers of $+1$ in $\boldsymbol{e}'$ is $m/2$, and the numbers of -1 in $\boldsymbol{e}'$ is 1. Additionally, we force the prover to prove that $\boldsymbol{e}'$ is in the form $\boldsymbol{e}' = \boldsymbol{e}_i \circ -\boldsymbol{i}_{i,l}$. To do so, Kawachi et al. divided a permutation π in Step P1 into two permutations.

Let π_h be a permutation over $[m]$ and π_t be a permutation over $[l]$. For a permutation π over $[m+l]$, we denote $\pi = \pi_h \odot \pi_t$ if

$$\pi = \begin{pmatrix} 1 & 2 & \cdots & m \\ \pi_h(1) & \pi_h(2) & \cdots & \pi_h(m) \end{pmatrix} \cdot \begin{pmatrix} m+1 & m+2 & \cdots & m+l \\ m+\pi_t(1) & m+\pi_t(2) & \cdots & m+\pi_t(l) \end{pmatrix}.$$

For any π_h and π_t, we have $(\pi_h \odot \pi_t)^{-1} = \pi_h^{-1} \odot \pi_t^{-1}$. For any $\boldsymbol{e}_h \in \mathbb{Z}^m$ and $\boldsymbol{e}_t \in \mathbb{Z}^l$, if $\pi = \pi_h \odot \pi_t$ then $\pi(\boldsymbol{e}_h \circ \boldsymbol{e}_t) = \pi_h(\boldsymbol{e}_h) \circ \pi_t(\boldsymbol{e}_t)$.

8.3.1 Description

We here construct an RID scheme $\mathsf{KTX\text{-}RID}_{\mathsf{GL},*}$ based on GapSVP. Similarly to the ID scheme $\mathsf{St\text{-}ID}^{+}_{\mathsf{GL},*}$ in Section 6.7, the protocol is repeated $t = \omega(\log n)$ times in parallel to achieve exponentially small soundness error. As in the previous section, we hide randomness in $\mathit{Com}_{\boldsymbol{A}}$.

Scheme 8.3.1 ($\mathsf{KTX\text{-}RID}_{\mathsf{GL,p}}$)**.** All the participants agree with the parameters $m = m(n)$, $q = q(n)$, and $w = w(n)$.

$\mathsf{Setup}(1^n)$**:** The same as Setup of the protocol in Section 6.7.

$\mathsf{Reg}(\boldsymbol{A}, i)$**:** The same as KG of the protocol in Section 6.7.

$\mathsf{RPKC}(\boldsymbol{A}, R = (\boldsymbol{u}_{i_1}, \ldots, \boldsymbol{u}_{i_l}))$**:** Output $\boldsymbol{A}' = [\boldsymbol{A}\, \boldsymbol{u}_{i_1}\, \ldots\, \boldsymbol{u}_{i_l}] \in \mathbb{Z}_q^{n\times(m+l)}$.

$\mathsf{RSKC}(\boldsymbol{A}, R = (\boldsymbol{u}_{i_1}, \ldots, \boldsymbol{u}_{i_l}), \boldsymbol{e}_{i_k})$**:** Output $\boldsymbol{e}' = \boldsymbol{e}_{i_k} \circ -\boldsymbol{i}_{k,l} \in \{0,1\}^m \times -\mathcal{S}(l, 1)$.

P **and** V**:** The common inputs are $\boldsymbol{A}$ and $(\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l)$. The prover's auxiliary input is $\boldsymbol{e}_i$ for some $i \in [l]$. Let $\boldsymbol{A}' := [\boldsymbol{A}\, \boldsymbol{u}_1\, \ldots\, \boldsymbol{u}_l]$ and $\boldsymbol{e} := \boldsymbol{e}_i \circ -\boldsymbol{i}_{i.l}$. We write Com instead of $\mathit{Com}_{\boldsymbol{A}}$ for ease of notation. Formally, they interact as follows:

Step P1: Choose random permutations π_h over $[m]$ and π_t over $[l]$. Let $\pi = \pi_h \odot \pi_t$. Choose a random vector $\boldsymbol{r} \in \mathbb{Z}_q^{m+l}$. Send commitments c_1, c_2, and c_3 as

- $c_1 = \mathit{Com}(\pi_h, \pi_t, \boldsymbol{A}'\boldsymbol{r})$,
- $c_2 = \mathit{Com}(\pi(\boldsymbol{r}))$,
- $c_3 = \mathit{Com}(\pi(\boldsymbol{e} + \boldsymbol{r}))$.

Step V1 Send a random challenge $ch \in \{1, 2, 3\}$ to P.

Step P2

- If $ch = 1$, reveal c_2 and c_3. Send $\boldsymbol{w} = \pi(\boldsymbol{e})$ and $\boldsymbol{x} = \pi(\boldsymbol{r})$.
- If $ch = 2$, reveal c_1 and c_2. Send $\phi_h = \pi_h$, $\phi_t = \pi_t$, and $\boldsymbol{y} = \boldsymbol{e} + \boldsymbol{r}$.
- If $ch = 3$, reveal c_1 and c_3. Send $\psi_h = \pi_h$, $\psi_t = \pi_t$, and $\boldsymbol{z} = \boldsymbol{r}$.

Step V2

- If $ch = 1$, check that $c_2 = \mathit{Com}(\boldsymbol{x})$, $c_3 = \mathit{Com}(\boldsymbol{w} + \boldsymbol{x})$, and $\boldsymbol{w}$ is in the form $\boldsymbol{w}_h \circ -\boldsymbol{i}_{j,l}$ for some j and $\boldsymbol{w}_h \in \mathcal{S}(m, w)$.

- If $ch = 2$, check that $c_1 = Com(\phi_h, \phi_t, \boldsymbol{A}'\boldsymbol{y})$ and $c_3 = Com((\phi_h \odot \phi_t)(\boldsymbol{y}))$.
- If $ch = 3$, check that $c_1 = Com(\psi_h, \psi_t, \boldsymbol{A}')$ and $c_2 = Com((\psi_h \odot \psi_t)(\boldsymbol{z}))$.

Output $dec = 1$ if all checks are passed, otherwise output $dec = 0$.

8.3.2 Security Proof

The security of the above protocol is stated as follows.

Theorem 8.3.2. *Let $m = m(n)$ and $q = q(n)$ be polynomially bounded functions satisfying the conditions that $m \geq 2(1+\delta)n\log q$ for some constant $\delta > 0$ and $q^n / |S(m, w)|$ is negligible in n. Assume that there exists an impersonator $\mathcal{A}$ that succeeds impersonation under concurrent chosen-group attack with non-negligible probability. Then there exists a probabilistic polynomial-time algorithm $\mathcal{A}$ that solves $\mathrm{SIS}^2_{q,m,\sqrt{m}}$.*

Combining Theorem 8.3.2 with Theorem 2.4.9, we obtain the following theorem.

Theorem 8.3.3. *For any $m(n) = \Theta(n\log n)$, there exist $q(n) = \mathrm{poly}(n)$, $w(n) = \omega(\log n)$, and $\gamma(n) = O(n\sqrt{\log n})$ such that $q^n / |S(m, w)|$ is negligible in n and the above scheme is secure against impersonation under concurrent chosen-group attack if SIVP^2_γ is hard in the worst case.*

The statistical anonymity of the above scheme follows from witness indistinguishability of the protocol.

Proof of Theorem 8.3.2. We will construct $\mathcal{A}$ solving $\mathrm{SIS}_{q,m,\sqrt{m}}$ with non-negligible probability by using an impersonator $\mathcal{I}$ which succeeds impersonation with non-negligible probability.

The algorithm $\mathcal{A}$, given input $\boldsymbol{A}$, feeds $\boldsymbol{A}$ to the impersonator $\mathcal{I}$. In the experiment, the impersonator $\mathcal{I}$ will call Init, Corr, and Prov. If $\mathcal{I}$ calls Init with input i, then $\mathcal{A}$ chooses $\boldsymbol{e}_i$ at random, computes $\boldsymbol{u}_i := \boldsymbol{A}\boldsymbol{e}_i$, and returns $\boldsymbol{u}_i$ to $\mathcal{I}$. $\mathcal{A}$ can simulate the oracles Corr and Prov, since $\mathcal{A}$ has the secret key $\boldsymbol{e}_i$ corresponding to the public key $\boldsymbol{u}_i$.

At the end of the experiment, $\mathcal{I}$ will impersonate the one in a ring $R = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l)$. Rewinding $\mathcal{I}$ three times, $\mathcal{A}$ obtains three valid transcripts as in the previous proof.

We next show how $\mathcal{A}$ obtains a secret key or finding a collision of the hash functions in the string commitment scheme by using three good transcripts. Assume that $\mathcal{A}$ has three transcripts $(cmt^{(i)}, ch^{(i)}, rsp^{(i)}, dec^{(i)})$ for $i = 1, 2, 3$ such that $cmt^{(1)} = cmt^{(2)} = cmt^{(3)}$, $dec^{(i)} = 1$ for all i, and $\{ch_j^{(1)}, ch_j^{(2)}, ch_j^{(3)}\} = \{1, 2, 3\}$ for some $j \in [t]$. Without loss of generality, we assume that $ch_j^{(i)} = i$. We parse $rsp_j^{(i)}$ as in Step V2. From the above argument, we have four equations as follows (We omit j for simplification):

$$\begin{array}{llll} c_1 & = Com_{\boldsymbol{A}}(\phi_h, \phi_t, \boldsymbol{A}'\boldsymbol{y}; \rho_1^{(2)}) & = Com_{\boldsymbol{A}}(\psi_h, \psi_t, \boldsymbol{A}'\boldsymbol{z}; \rho_1^{(3)}), \\ c_2 & = Com_{\boldsymbol{A}}(\boldsymbol{x}; \rho_2^{(1)}) & = Com_{\boldsymbol{A}}((\psi_h \odot \psi_t)(\boldsymbol{z}); \rho_2^{(3)}), \\ c_3 & = Com_{\boldsymbol{A}}(\boldsymbol{w} + \boldsymbol{x}; \rho_3^{(1)}) & = Com_{\boldsymbol{A}}((\phi_h \odot \phi_t)(\boldsymbol{y}); \rho_3^{(2)}), \\ \boldsymbol{w} & = \boldsymbol{w}_h \circ -\boldsymbol{i}_{k,l} \text{ for some } k & \text{and } \boldsymbol{w}_h \in \mathcal{S}(m, w). \end{array}$$

If there exists a distinct pair of arguments of $Com_{\boldsymbol{A}}$, $\mathcal{A}$ obtains a collision for $\boldsymbol{A}$ and solves $\mathrm{SIS}_{q,m,\sqrt{m}}$.

Let us assume that there exist no distinct pairs. Let π be an inverse permutation of $\phi_h \odot \phi_t$. From the first equation, we obtain the equation $\pi^{-1} = \phi_h \odot \phi_t = \psi_h \odot \psi_t$. Combining with the third equation, we have $\boldsymbol{y} = \pi(\boldsymbol{w} + \boldsymbol{x})$. Thus, we have $\boldsymbol{A}'\boldsymbol{z} = \boldsymbol{A}'(\pi(\boldsymbol{w}) + \pi(\boldsymbol{x}))$. From the second equation, $\boldsymbol{z} = \pi(\boldsymbol{x})$. Hence, we obtain $\boldsymbol{A}' \cdot \pi(\boldsymbol{w}) = \boldsymbol{0}$. We have $\pi = \pi_h \odot \pi_t$ for some permutations π_h and π_t over $[m]$ and $[l]$ respectively, since π is inverse of $\phi_h \odot \phi_t$. Thus, we have $\boldsymbol{A}'(\pi_h(\boldsymbol{w}_h) \circ \pi_t(-\boldsymbol{i}_{k.l})) = \boldsymbol{0}$. That is $\boldsymbol{u}_{\pi_t(k)} = \boldsymbol{A}\pi_h(\boldsymbol{w}_h)$. By using same argument in the previous proof, we have that $\pi_h(\boldsymbol{w}_h) \neq \boldsymbol{e}_{\pi_t(k)}$ with probability at least $1/2 - \mathsf{negl}(n)$. So, $\mathcal{A}$ outputs $\boldsymbol{z} = \boldsymbol{e}_{\pi_t(k)} - \pi_h(\boldsymbol{w}_h)$ as a solution for $\mathrm{SIS}_{q,m,\sqrt{m}}$.

□

8.3.3 The Cyclic/Ideal version

Changing the key-generation algorithm, we have a lightweight version $\mathsf{KTX\text{-}RID}_{\mathsf{C/IL},*}$. For simplicity, we fix $\mathbf{f} = x^{2^k} + 1$.

Theorem 8.3.4. *Let m, q, and w be polynomially bounded functions of n such that $m > 4\log q$, q is a prime with $q \equiv 3 \pmod 8$, and $q^n / |\mathcal{S}(mn, w)|$ is negligible in n. Then, if $\mathbf{f}\text{-SIS}^{\infty}_{q,m,1}$ is hard on average, the ID scheme $\mathsf{KTX\text{-}RID}_{\mathsf{C/IL},*}$ is concurrently secure.*

In addition, let $m = m(n)$, $q = q(n)$, and $w = w(n)$ be polynomially bounded functions such that $q > 6mn^{3/2}\log n$, and $q^n / |\mathcal{S}(mn, w)|$ is negligible in n. Then for $\gamma = 72mn\log^2 n$, if $\mathbf{f}\text{-SVP}^{\infty}_{\gamma}$ is hard in the worst case then the ID scheme $\mathsf{KTX\text{-}RID}_{\mathsf{C/IL},*}$ is concurrently secure.

sketch. We show that if there exists an impersonator $\mathcal{I}$ which succeeds impersonation under concurrent chosen-group attack with non-negligible probability, there exists $\mathcal{A}$ that finds a collision (z_1, z_2) for $h_{\check{\boldsymbol{a}}}$.

The algorithm $\mathcal{A}$, given input $\check{\boldsymbol{a}} \in R^m_{\mathbf{f},q}$, feeds $\check{\boldsymbol{a}}$ to the impersonator $\mathcal{I}$. In the experiment, the impersonator $\mathcal{I}$ will call Init, Corr, and Prov. If $\mathcal{I}$ calls Init with input i, then $\mathcal{A}$ chooses $\check{\boldsymbol{e}}_i \in \mathcal{S}(mn, w)$ at random, computes $\mathbf{u}_i := h_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}}_i)$, and returns $\mathbf{u}_i$ to $\mathcal{I}$. $\mathcal{A}$ can correctly simulate the oracles Corr and Prov, since $\mathcal{A}$ has the secret key $\check{\boldsymbol{e}}_i$ corresponding to the public key $\mathbf{u}_i$.

At the end of the experiment, $\mathcal{I}$ will impersonate the one of a ring $R = (\mathbf{u}_1, \ldots, \mathbf{u}_l)$. Rewinding $\mathcal{A}$ three times, $\mathcal{A}$ obtains $(s, \rho) \neq (s', \rho')$ such that $Com_{\boldsymbol{A}}(s; \rho) = Com_{\boldsymbol{A}}(s'; \rho')$ or a vector $\boldsymbol{e} = \boldsymbol{e}_h \circ \boldsymbol{e}_t$ such that $[\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})\, \boldsymbol{u}_1 \ldots \boldsymbol{u}_l]\boldsymbol{e} = \boldsymbol{0}$, where $\boldsymbol{x}_h \in \{0,1\}^{mn}$, $\boldsymbol{e}_t = -\boldsymbol{i}_{k,l}$ for some k, and $\boldsymbol{e}_h \in \mathcal{S}(mn, w)$ as in the proofs of Theorem 6.7.3 and Theorem 8.3.2.

In the former case, $\mathcal{A}$ computes $\boldsymbol{z} \neq \boldsymbol{z}' \in \{0,1\}^{mn}$ such that $Com_{\check{\boldsymbol{a}}}(s;\rho) = \mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})\boldsymbol{z}$ and $Com_{\check{\boldsymbol{a}}}(s';\rho') = \mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})\boldsymbol{z}'$. Hence, $\mathcal{A}$ outputs $(\boldsymbol{z}, \boldsymbol{z}')$ as a collision for $h_{\check{\boldsymbol{a}}}$.

In the latter case, we have $\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}}) \cdot \boldsymbol{e}_h = \boldsymbol{u}_k$. By the same argument as in the proof of Theorem 8.3.2, we have that $\boldsymbol{e}_h \neq \boldsymbol{e}_k$ with probability at least $1/2$. Hence, $\mathcal{A}$ outputs $(\boldsymbol{e}_h, \boldsymbol{e}_k)$ as a collision for $h_{\check{\boldsymbol{a}}}$. □

9 Interlude: Zero-Knowledge Protocols on NTRU

As an interlude, we review zero-knowledge and proof-of-knowledge protocols for NTRU by Xagawa and Tanaka [XT09] which exploited Stern's ID scheme. One is for the relation on secret-key knowledge and the other for that on plaintext knowledge. They are the first non-trivial constructions of these protocols for NTRU. Additionally, the former directly yields an identification scheme based on NTRU.

Organization: In Section 9.1, we review background of the NTRU encryption scheme. Section 9.2 gives a brief review of the NTRU encryption scheme. We give exiting relations between NTRU and lattices in Section 9.3. Section 9.4 reviews the Xagawa–Tanaka protocol for the basic relations. Section 9.5 gives a detail of ID scheme obtained from the XT protocol. Section 9.6 compares several ID schemes based on combinatorial problems. Section 9.7 gives some concluding remarks on the protocol and the ID scheme.

9.1 Introduction

Background: In 1996, Hoffstein, Pipher, and Silverman proposed a public-key encryption system, NTRU [HPS98] (the conference version is appeared in ANTS III [HPS98]). The main attractions of this encryption scheme are fast key-generation, encryption, and decryption, and compact sizes of keys. Other lattice-based encryption schemes, such as the Ajtai–Dwork cryptosystem [AD97], the GGH cryptosystem [GGH97b], and the Regev cryptosystems [Reg03, Reg09] do not have all of these attractions. (See Chapter 12 for the details of them.) Additionally, it seems bearable against threat of quantum computers. After proposing the scheme, they founded a company named NTRU Cryptosystems.

The proposers of NTRU have modified the parameters of the system. As instantiations of NTRU, there are NTRU-1998 [HPS98], NTRU-2001 [HS00], NTRU-2005 [HGSW05], NTRU-2007 [HHGP+07], and NTRU-2008 [WHGH+08, HHHGW09], where the last instantiation is in IEEE P1363.1/D12. There are several attacks after the Coppersmith–Shamir attack [CS97]. For chosen plaintext attack, see Coppersmith and Shamir [CS97], Odlyzko's meet-in-the-middle attack [HGSW03], and Howgrave-Graham [HG07]. For chosen ciphertext attack, see [JJ00, HHHK03, HGNP+03, MR06, GN07]. For the summary of the attacks, see, e.g., Mol and Yung [MY08].

While approximately forty papers have dealt with NTRU, surprisingly, there are no *protocols* except these for encryption or signature. For example, there are no secure identification schemes based on the NTRU problems and proofs of plaintext knowledge for NTRU. This contrasts with the situation that the number-theoretical assumptions allow us to construct concurrent-secure identification schemes and non-malleable proofs of plaintext knowledge for the RSA, Rabin, Paillier, and El-Gamal encryption schemes [Kat03].

Techniques: The main idea of Xagawa and Tanaka [XT09] is plugging the structure of NTRU into a variant of Stern's protocol [Ste96, KTX08].

Kawachi, Tanaka, and Xagawa [KTX08] observed that Stern's protocol can be used for the relations on q-ary lattices. The relation is the set of $((\boldsymbol{A}, \boldsymbol{u}), \boldsymbol{e}) \in (\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n) \times \{0,1\}^m$ such that $\boldsymbol{A}\boldsymbol{e} \equiv \boldsymbol{u} \pmod q$ and the Hamming weight of $\boldsymbol{e}$ is d (see Section 6.6 and Section 6.7).

In addition, note that the well-known NTRU lattice is indeed q-ary lattice, which has a representation $\Lambda_q^{\perp}(\boldsymbol{A}) = \{\boldsymbol{e} \in \mathbb{Z}^m \mid \boldsymbol{A}\boldsymbol{e} \equiv \boldsymbol{0} \pmod q\}$ (see [CS97, MR08]).

However, there are some difficulties to connect NTRU with the Stern's protocol directly. In order to connect NTRU with the variant of Stern's protocol, we modify the structure of ad hoc anonymous identification schemes by Kawachi et al. [KTX08], which introduced the permutation splitting technique in Stern's protocol, rather than the identification scheme by them. By this modification, we pattern a statistical-zero-knowledge and proof-of-knowledge argument for the generalized relations, say, the set of polynomials $((\mathbf{a}, \mathbf{b}, \mathbf{z}), (\mathbf{x}, \mathbf{u}))$ such that $\mathbf{a} \otimes \mathbf{x} + \mathbf{b} \otimes \mathbf{u} \equiv \mathbf{z} \pmod{x^n - 1, q}$ and *each* Hamming weight of $\mathbf{x}$ and $\mathbf{u}$ is d_x and d_y, respectively. Then, we modify the protocols in order to employ the relations on secret-key knowledge and plaintext knowledge tailored for each instantiation of NTRU.

Related works: It is well-known that the existence of one-way functions implies computational-zero-knowledge proof systems for any NP-relation. However, this general proof system is less efficient than the arguments by Xagawa and Tanaka [XT09].

In general, one has a simple challenge-and-response protocol for the relation on secret-key knowledge: The verifier sends a random ciphertext to the prover, and the prover answers the plaintext of the ciphertext. This simple protocol is honest-verifier perfect zero knowledge for the relation on secret-key knowledge. However, a malicious verifier can use this protocol as the decryption oracle. In some instantiations of NTRU, this yields the universal break [MY08].

One can consider a protocol by combining the observations by Coppersmith and Shamir mentioned above with statistical-zero-knowledge proof systems for GapSVP and GapCVP by Micciancio and Vadhan [MV03] (Section 6.3). This combination might fit for our purpose, however, we cannot provide reasonable analysis of it.

Recent studies on lattice-based cryptography gave several protocols for lattice problems and cryptographic primitives based on the worst-case hardness of lattice problems. There are statistical zero-knowledge proof systems for coGapSVP and coGapCVP by Goldreich and Goldwasser [GG00], and ones with efficient prover for GapSVP and GapCVP by Micciancio and Vadhan [MV03]. Recently, Peikert and Vaikuntanathan gave non-interactive statistical-zero-knowledge proof systems with efficient prover for several lattice problems [PV08]. Lyubashevsky [Lyu08a, Lyu09] (Section 6.5 and Section 6.8) and Kawachi et al. [KTX08] (Section 6.7) proposed concurrently-secure lattice-based identification schemes. It will be an interesting task to construct concurrently-secure ID scheme and non-interactive zero-knowledge protocols for NTRU.

We next discuss the plaintext knowledge. The Xagawa–Tanaka proof-of-knowledge arguments for the relations on plaintext knowledge are related to proof of plaintext knowledge (PPK). Explicit formalization of PPK is due to Aumann and Rabin (cited in Katz [Kat03]). According to Katz [Kat03, Section 1.2], they gave a generic solution for any public-key encryption scheme and their protocol is honest-verifier zero knowledge, while the arguments are (cheating-verifier) statistical zero knowledge.

There are many identification schemes based on the combinatorial problems; Shamir's scheme [Sha89] based on the permuted kernel problem, Stern's scheme [Ste96] based on the syndrome decoding problem, and the scheme by Pointcheval and Poupard [PP03] based on the permuted perceptron problem, and etc. Section 9.6 compares the NTRU-based ID scheme and the identification schemes mentioned above.

Finally, we report that we found an independent work by Gaborit and Girault [GG07]. They proposed light-weight variants of Stern's identification scheme by using NTRU-like codes, double circulant linear codes and assumed the hardness of the syndrome decoding problem of their NTRU-like codes. We note that their paper lacks the proof of security and does not show zero-knowledge property of the protocol. (We also note that we can easily repair the protocol and obtain their security proof.) We also note that their protocol cannot be used as a proof-of-knowledge argument for the relation on secret-key knowledge since they did not split the permutation.

9.2 Brief Sketch of NTRU

In this section, we briefly review NTRU. For details, see the original paper [HPS98], the proposals of the parameters [HS00, HGSW05, HHGP$^+$07, WHGH$^+$08, HHHGW09], and the description in Section 12.6

For a positive integer n, NTRU is defined on a quotient ring $R = \mathbb{Z}[x]/(x^n - 1)$. For a positive integer q, we denote $\mathbb{Z}[x]/(q, x^n - 1)$ by R_q. We identify R with $\mathbb{Z}^n$ by identifying $\mathbf{f} = \sum_{i=0}^{n-1} f_i x^i \in R$ with $\mathbf{f} = (f_0, \dots, f_{n-1}) \in \mathbb{Z}^n$. We also identify R_q with $\mathbb{Z}_q^n$.

Intuitively, the security is based on the hardness to factor a product of two short polynomials in R_q. Let n denote the dimension of R_q. The subsets of R_q, $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_m$, $\mathcal{L}_r$, and $\mathcal{L}_F$ are defined later. They are used for key generation, encryption, and decryption. While we do not consider the decryption in this paper, we note the decryption procedure.

Scheme 9.2.1 (NTRUEncrypt)**.** Let n denote the dimension of R_q. All the participant agree the parameters settings.

Setup(1^n)**:** Given the security parameter n, output 1^n.

KeyGen($param = 1^n$)**:** Choose $\mathbf{f} \leftarrow \mathcal{L}_f$ and $\mathbf{g} \leftarrow \mathcal{L}_g$ with the constrain that $\mathbf{f}$ is invertible in R_q and R_p. Set $\mathbf{F}_q \leftarrow \mathbf{f}^{-1}$ in R_q. Compute $\mathbf{h} \leftarrow p \otimes \mathbf{g} \otimes \mathbf{F}_q$ in R_q. The public key is $\mathbf{h}$ and the secret key is $\mathbf{f}$.

Enc($ek = \mathbf{h}, msg = \mathbf{m}$)**:** The plaintext is $\mathbf{m} \in \mathcal{L}_m$. Generate a random polynomial $\mathbf{r} \leftarrow \mathcal{L}_r$ and compute $\mathbf{c} \leftarrow \mathbf{h} \otimes \mathbf{r} + \mathbf{m}$ in R_q. The ciphertext is $\mathbf{c}$.

Dec($dk = \mathbf{f}, ct = \mathbf{c}$)**:** The ciphertext is $\mathbf{c} \in R_q$. Compute $\mathbf{a}' \leftarrow \mathbf{f} \otimes \mathbf{c}$ in R_q. Compute $\mathbf{a} \leftarrow p \otimes \mathbf{g} \otimes \mathbf{r} + \mathbf{f} \otimes \mathbf{m}$ in R from $\mathbf{a}'$ by using a centering algorithm. Compute $\mathbf{F}_p \leftarrow \mathbf{f}^{-1}$ in R_p. Compute $\mathbf{m}' \leftarrow \mathbf{F}_p \otimes \mathbf{a}$ in R_p. The obtained plaintext is $\mathbf{m}'$.

The decryption correctly works since the parameters are chosen carefully to ensure that $\mathbf{a} = p \otimes \mathbf{g} \otimes \mathbf{r} + \mathbf{f} \otimes \mathbf{m}$ in R with high probability. We omit the details of the parameter setting; see the original paper or the papers on instantiations [HPS98, WHGH$^+$08, HHHGW09].

Let $\mathcal{T}$ denote $\{-1, 0, +1\}^n$. $\mathcal{T}(d_1, d_2)$ denotes the subset of $\mathcal{T}$ such that each polynomial in $\mathcal{T}(d_1, d_2)$ has exactly d_1 coefficients set to 1 and d_2 coefficients set to -1. For an integer a and a subset $\mathcal{S} \subseteq R_q$, we define $a\mathcal{S}$ as $\{a\mathbf{f} : \mathbf{f} \in \mathcal{S}\}$. For a subset $\mathcal{S} \subseteq R_q$, $\mathcal{S}^*$ denotes the set of the polynomials in $\mathcal{S}$ which have the inverses in R_q, i.e., $\mathcal{S}^* = \{\mathbf{f} \in \mathcal{S} : \exists \mathbf{f}^{-1} \in R_q\}$.

There are five instantiations of NTRU, NTRU-1998 [HPS98], NTRU-2001 [HS00], NTRU-2005 [HGSW05], NTRU-2007 [HHGP$^+$07], and NTRU-2008 [WHGH$^+$08, HHHGW09], which are summarized in Table 9.1. The following table summarizes the parameter sets of these instantiations. In Table 9.1, we use $\mathcal{T}(d_g, d_g)$ instead of $\mathcal{T}(d_g, d_g)^*$ in NTRU-2008 for certain technical reason.

Parameter Sets	q	p	$\mathcal{L}_f$	$\mathcal{L}_g$	$\mathcal{L}_m$	$\mathcal{L}_r$	$\mathcal{L}_F$
NTRU-1998	2^k	3	$\mathcal{T}(d_f, d_f - 1)^*$	$\mathcal{T}(d_g, d_g)$	$\mathcal{T}$	$\mathcal{T}(d_r, d_r)$	-
NTRU-2001	prime	$2 + x$	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{B}(d_g)$	$\mathcal{B}$	$\mathcal{B}(d_r)$	$\mathcal{B}(d_F)$
NTRU-2005	prime	2	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{B}(N/2)^*$	$\mathcal{B}$	$\mathcal{X}(d_r)$	$\mathcal{X}(d_F)$
NTRU-2007	2^k	3	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{T}(d_f, d_f - 1)^*$	$\mathcal{T}(d_f, d_f - 1)$	$\mathcal{T}(d_f, d_f - 1)$	$\mathcal{T}(d_f, d_f - 1)$
NTRU-2008	2^k	3	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{T}(d_g, d_g)$	$\mathcal{T}$	$\mathcal{T}(d_r, d_r)$	$\mathcal{T}(d_F)$

Table 9.1: Parameter sets. In NTRU-1998, $\mathbf{f}$ must be invertible in R_p.

9.3 Interpretation of NTRU as Lattice-based Encryption

Since we connect NTRU and the lattice-based protocol, we briefly review NTRU lattices.

NTRU lattices: We consider the following matrix $\boldsymbol{C}$ which is generated by a secret key:

$$\boldsymbol{C} = \left[\mathrm{Rot}^T(\mathbf{f}) \quad \mathrm{Rot}^T(p \otimes \mathbf{g})\right].$$

An NTRU lattice [CS97] is generated by a basis

$$\boldsymbol{H} = \begin{bmatrix} \mathrm{Rot}(1) & \mathrm{Rot}(0) \\ \mathrm{Rot}(\mathbf{h}) & \mathrm{Rot}(q) \end{bmatrix}.$$

It is easy to verify that $L(\boldsymbol{H}) = \Lambda_q(\boldsymbol{C})$ by the equation $\mathbf{h} \equiv \mathbf{f}^{-1} \otimes (p \otimes \mathbf{g}) \pmod{q}$.

Since Stern's protocol and its variant used $\Lambda_q^{\perp}(\boldsymbol{A})$ for some $\boldsymbol{A}$ rather than $\Lambda_q(\boldsymbol{C})$, we have to find $\boldsymbol{A} \in \mathbb{Z}_q^{n \times 2n}$ such that $L(\boldsymbol{H}) = \Lambda_q^{\perp}(\boldsymbol{A})$. As noted in the paper proposing NTRUSign [HHGP+03], we can verify that

$$L(\boldsymbol{H}) = \Lambda_q^{\perp}([-\mathrm{Rot}(\mathbf{h})\ \mathrm{Rot}(1)]).$$

Thus, we define $\boldsymbol{A} = [-\mathrm{Rot}(\mathbf{h})\ \mathrm{Rot}(1)]$. In the following, we mainly consider NTRU lattices in this form. We will give the details in Section 12.6.

9.4 The Xagawa–Tanaka Protocol

Now, we review the Xagawa–Tanaka protocol [XT09]. We first quickly review the Stern protocol (Section 6.6) and the KTX RID protocol (Section 8.3).

9.4.1 Relations of Stern's Protocol and its Variant

Let $\mathcal{B}_m(d)$ denote the set of m-dimensional binary vectors whose Hamming weights are d, i.e., the numbers of 1's are exactly d. As already seen in Section 6.6, his protocol is for the following relation:

$$\{((\boldsymbol{A}, \boldsymbol{u}), \boldsymbol{e}) \in (\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^n) \times \{0,1\}^m : (\boldsymbol{A}\boldsymbol{e} \equiv \boldsymbol{u} \pmod{q}) \wedge (\boldsymbol{e} \in \mathcal{B}_m(d))\}.$$

Kawachi et al. proposed the variants of Stern's protocol [KTX08] which appeared in Section 8.3. In a ring identification scheme based on their variant, the protocol is for the relation

$$\{((\boldsymbol{A}, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_l), \boldsymbol{e}) \in (\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times l}) \times \{-1, 0, +1\}^{m+l}$$
$$: (\boldsymbol{e} = \boldsymbol{e}_h \circ \boldsymbol{e}_t) \wedge (\boldsymbol{e}_h \in \mathcal{B}_m(d)) \wedge (\boldsymbol{e}_t \in -\mathcal{B}_l(1)) \wedge (\boldsymbol{A}\boldsymbol{e}_h + [\boldsymbol{u}_1 \ \ldots \ \boldsymbol{u}_l]\boldsymbol{e}_l \equiv \boldsymbol{0} \pmod{q})\}.$$

In order to design the protocol, they split a permutation in Stern's protocol; they put two permutations in their protocol, while Stern put one permutation. (See Section 8.3 and [KTX08, Section 5].)

9.4.2 The Xagawa–Tanaka Protocol

The Xagawa–Tanaka protocol [XT09] has a similar structure of the ring identification schemes by Kawachi et al. [KTX08], which are obtained by splitting a permutation in Stern's protocol [Ste96] as in the above. We replace the two set, $\mathcal{B}_m(m/2)$ and $-\mathcal{B}_l(1)$, with the following enumeration sets.

Enumeration sets: For a positive integer n, we denote by $[n]$ the set $\{0, \ldots, n-1\}$. S_n denotes the n-dimensional permutation group, i.e., the group consisting of all of the permutations over $[n]$. The operator $*$ means the composition of permutations, that is, $(\pi * \phi)(\mathbf{x}) = \pi(\phi(\mathbf{x}))$.

We define a property of a subset of R_q. Let π be a permutation over $[n]$. For an n-dimensional vector $\boldsymbol{f} = (f_0, \ldots, f_{n-1}) \in \mathbb{Z}_q^n$, we define $\pi(\boldsymbol{f}) = (f_{\pi(0)}, \ldots, f_{\pi(n-1)})$. We note that, for a permutation π over $[n]$ and two polynomials $\mathbf{a}$ and $\mathbf{b}$ in R_q, $\pi(\mathbf{a} + \mathbf{b}) = \pi(\mathbf{a}) + \pi(\mathbf{b})$. For a polynomial $\boldsymbol{x} \in R_q$, $\mathcal{S}_{\boldsymbol{x}}$ denotes $\{\pi(\boldsymbol{x}) : \pi \in S_n\} \subset R_q$. We call these sets *enumeration sets*. We note that $\mathcal{T}(d_1, d_2)$ is an enumeration set, while $\mathcal{T}$ is not.

9.4.3 Description

For two enumeration sets $\mathcal{S}_h$ and $\mathcal{S}_t$, consider the following relation:

$$R = \{((\mathrm{Rot}(\mathbf{a}_h), \mathrm{Rot}(\mathbf{a}_t), \mathbf{u}), (\boldsymbol{e}_h, \boldsymbol{e}_t)) \in (\mathcal{M}_q \times \mathcal{M}_q \times \mathbb{Z}_q^n) \times (\mathbb{Z}_q^n \times \mathbb{Z}_q^n)$$
$$: (\mathrm{Rot}(\mathbf{a}_h) \cdot \mathbf{e}_h + \mathrm{Rot}(\mathbf{a}_t) \cdot \mathbf{e}_t = \mathbf{u}) \wedge (\mathbf{e}_h \in \mathcal{S}_h) \wedge (\mathbf{e}_t \in \mathcal{S}_t)\}.$$

This relation is interpreted as follows:

$$R = \{((\mathbf{a}_h, \mathbf{a}_t, \mathbf{u}), (\mathbf{e}_h, \mathbf{e}_t)) \in R_q^3 \times R_q^2 : (\mathbf{a}_h \otimes \mathbf{e}_h + \mathbf{a}_t \otimes \mathbf{e}_t = \mathbf{u}) \wedge (\mathbf{e}_h \in \mathcal{S}_h) \wedge (\mathbf{e}_t \in \mathcal{S}_t)\}.$$

If $\mathbf{a}_h = -\mathbf{h}$ and $\mathbf{a}_t = 1$, the relation R is directly for the NTRU lattice.

Let *Com* be the special type of a statistically-hiding and computationally-binding string-commitment scheme as in Section 5.1.3.

Now, we describe the XT protocol, which is statistical zero knowledge and proof of knowledge for the relation R. For ease of notation, we do not write the randomness of the function *Com*.

Scheme 9.4.1 (The Xagawa–Tanaka protocol [XT09])**.** The common input is a triplet $(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u}) \in R_q^3$. Prover's auxiliary inputs are $\mathbf{e}_h$ and $\mathbf{e}_t$ such that $\mathbf{a}_h \otimes \mathbf{e}_h + \mathbf{a}_t \otimes \mathbf{e}_t = \mathbf{u}$, $\mathbf{e}_h \in \mathcal{S}_h$, and $\mathbf{e}_t \in \mathcal{S}_t$. The prover P and the verifier V interact as follows:

Step P1: Choose random permutations π_h and π_t over $[n]$. Choose random vectors $\mathbf{r}_h$ and $\mathbf{r}_t \in R_q$ and send commitments $y = (c_1, c_2, c_3)$ computed as follows:

- $c_1 \leftarrow Com(\pi_h, \pi_t, \mathbf{a}_h \otimes \mathbf{r}_h + \mathbf{a}_t \otimes \mathbf{r}_t)$,
- $c_2 \leftarrow Com(\pi_h(\mathbf{r}_h), \pi_t(\mathbf{r}_t))$,
- $c_3 \leftarrow Com(\pi_h(\mathbf{e}_h + \mathbf{r}_h), \pi_t(\mathbf{e}_t + \mathbf{r}_t))$.

Step V1: Send a random challenge $ch \in \{1, 2, 3\}$ to P. Each challenge 1, 2, and 3 corresponds to verifier's check "permuted," "masked," and "random."

Step P2:

- If $ch = 1$, it reveals c_2 and c_3. So, send $z_1 = (\mathbf{w}_h, \mathbf{w}_t, \mathbf{x}_h, \mathbf{x}_t) \leftarrow (\pi_h(\mathbf{e}_h), \pi_t(\mathbf{e}_t), \pi_h(\mathbf{r}_h), \pi_t(\mathbf{r}_t))$.
- If $ch = 2$, it reveals c_1 and c_3. Send $z_2 = (\phi_h, \phi_t, \mathbf{y}_h, \mathbf{y}_t) \leftarrow (\pi_h, \pi_t, \mathbf{e}_h + \mathbf{r}_h, \mathbf{e}_t + \mathbf{r}_t)$.
- If $ch = 3$, it reveals c_1 and c_2. Send $(\psi_h, \psi_t, \mathbf{z}_h, \mathbf{z}_t) \leftarrow (\pi_h, \pi_t, \mathbf{r}_h, \mathbf{r}_t)$.

Step V2:

- If $ch = 1$, check that $c_2 = Com(\mathbf{x}_h, \mathbf{x}_t)$, $c_3 = Com(\mathbf{w}_h + \mathbf{x}_h, \mathbf{w}_t + \mathbf{x}_t)$, $\mathbf{w}_h \in \mathcal{S}_h$, and $\mathbf{w}_t \in \mathcal{S}_t$.
- If $ch = 2$, check that $c_1 = Com(\phi_h, \phi_t, \mathbf{a}_h \otimes \mathbf{y}_h + \mathbf{a}_t \otimes \mathbf{y}_t - \mathbf{u})$, $c_3 = Com(\phi_h(\mathbf{y}_h), \phi_t(\mathbf{y}_t))$.
- If $ch = 3$, check that $c_1 = Com(\psi_h, \psi_t, \mathbf{a}_h \otimes \mathbf{z}_h + \mathbf{a}_t \otimes \mathbf{z}_t)$ and $c_2 = Com(\psi_h(\mathbf{z}_h), \psi_t(\mathbf{z}_t))$.

Output $dec = 1$ if all of the above checks are passed, otherwise output $dec = 0$.

Theorem 9.4.2. *If Com is a statistically-hiding and computationally-binding string-commitment scheme, the above protocol is a statistical-zero-knowledge and proof-of-knowledge argument for a relation R with soundness* $2/3$.

Proof. The correctness of the above protocol can easily be shown. The existence of a knowledge extractor implies the soundness of the protocol. Thus, in the proof, we show the existence of a simulator and a knowledge extractor. However, since these proofs are very similar to the ones in [Ste96, KTX08], we omit the details and give the sketch of the proof.

Statistical zero knowledge: The construction of the simulator is similar to the ones in [Ste96] and [KTX08].

We construct a simulator S which, on input $(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u})$ and given oracle access to a cheating verifier V^*, outputs a simulated transcript. S chooses a random value $\overline{ch}$ from $\{1, 2, 3\}$, a prediction of the value V^* *will not* choose. Next, the simulator chooses a random tape r' of V^*. We only show how the simulator works in the case $\overline{ch} = 1$. The remaining cases can be proved by the similar way to the proof of

Lemma 6.7.2.

Case $\overline{ch} = 1$: S computes $\mathbf{e}'_h$ and $\mathbf{e}'_t$ such that $\mathbf{a}_h \otimes \mathbf{e}'_h + \mathbf{a}_t \otimes \mathbf{e}'_t = \mathbf{u}$ by using linear algebra. Next, it chooses random permutations π'_h and π'_t over $[n]$, random polynomials $\mathbf{r}'_h$ and $\mathbf{r}'_t$ from R_q, and random strings ρ'_i for $i = 1, 2, 3$. It computes commitments as

- $c'_1 = Com(\pi'_h, \pi'_t, \mathbf{a}_h \otimes \mathbf{r}'_h + \mathbf{a}_t \otimes \mathbf{r}'_t; \rho'_1)$,
- $c'_2 = Com(\pi'_h(\mathbf{r}'_h), \pi'_t(\mathbf{r}'_t); \rho'_2)$,
- $c'_3 = Com(\pi'_h(\mathbf{e}'_h + \mathbf{r}'_h), \pi'_t(\mathbf{e}'_t + \mathbf{r}'_t); \rho'_3)$.

It sends the commitments to V^* and receives a challenge ch from V^*. The simulator S computes a transcript as follows:

- If $ch = 1$, it outputs $\perp$ and halts.
- If $ch = 2$, it outputs $(r'; (c'_1, c'_2, c'_3), 2, (\pi'_h, \pi'_t, \mathbf{e}'_h + \mathbf{r}'_h, \mathbf{e}'_t + \mathbf{r}'_t, \rho'_1, \rho'_3))$.
- If $ch = 3$, it outputs $(r'; (c'_1, c'_2, c'_3), 3, (\pi'_h, \pi'_t, \mathbf{r}'_h, \mathbf{r}'_t, \rho'_1, \rho'_2))$.

We start the analysis of the case $ch = 2$. In this case, we have that

$$\begin{aligned} \mathrm{View}^{\mathsf{P}}_{\mathsf{V}^*(z)}(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u}) &= (r; (c_1, c_2, c_3), 2, (\pi_h, \pi_t, \mathbf{e}_h + \mathbf{r}_h, \mathbf{e}_t + \mathbf{r}_t, \rho_1, \rho_3)), \\ \mathsf{S}(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u}) &= (r'; (c'_1, c'_2, c'_3), 2, (\pi'_h, \pi'_t, \mathbf{e}'_h + \mathbf{r}'_h, \mathbf{e}'_t + \mathbf{r}'_t, \rho'_1, \rho'_3)). \end{aligned}$$

Consider a one-to-one mapping $(\pi'_h, \pi'_t, \mathbf{r}'_h, \mathbf{r}'_t, \rho'_1, \rho'_3) = (\pi_h, \pi_t, \mathbf{r}_h + \mathbf{e}_h - \mathbf{e}'_h, \mathbf{r}_t + \mathbf{e}_t - \mathbf{e}'_t, \rho_1, \rho_3)$. By this equation, we have that $c'_1 = c_1$ and $c'_3 = c_3$, and the responses from the simulator equal to the ones from the prover. From the statistically-hiding property of *Com*, the statistical distance between the distributions of c_2 and c'_2 is negligible. Thus, the distributions of $\mathrm{View}^{\mathsf{P}}_{\mathsf{V}^*(z)}(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u})$ and $\mathsf{S}(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u})$ are statistically close.

In the case $ch = 3$, it is easy to verify the statistical distance by setting $(\pi'_h, \pi'_t, \mathbf{r}'_h, \mathbf{r}'_t, \rho'_1, \rho'_2) = (\pi_h, \pi_t, \mathbf{r}_h, \mathbf{r}_t, \rho_1, \rho_2)$.

Proof of knowledge: We construct the extractor following [Ste96] and [KTX08]. Assume that there exists P^* that convincing V with probability $2/3 + \epsilon$. The knowledge extractor K works as follows:

1. Choose a random tape of P^*.
2. Obtain three transcripts by acting as the verifier and by setting $ch = 1, 2, 3$. Each $\rho_i^{(j)}$ denotes the random string in the commitment c_i in the case that $ch = j$. The three transcripts are $((c_1, c_2, c_3), 1, (\mathbf{w}_h, \mathbf{w}_t, \mathbf{x}_h, \mathbf{x}_t, \rho_2^{(1)}, \rho_3^{(1)}))$, $((c_1, c_2, c_3), 2, (\phi_h, \phi_t, \mathbf{y}_h, \mathbf{y}_t, \rho_1^{(2)}, \rho_3^{(2)}))$, and $((c_1, c_2, c_3), 3, (\psi_h, \psi_t, \mathbf{z}_h, \mathbf{z}_t, \rho_1^{(3)}, \rho_2^{(3)}))$.
3. Output $(\phi_h^{-1}(\mathbf{w}_h), \phi_t^{-1}(\mathbf{w}_t))$ as a witness.

We analyze the probability that the output is the witness corresponding to $(\mathbf{a}_h, \mathbf{a}_t, \mathbf{u})$. Since the probability that P^* convincing V is $2/3 + \epsilon$, for ϵ fraction of random tapes three transcripts are valid. Hence, we have the following equations:

$$Com(\phi_h, \phi_t, \mathbf{a}_h \otimes \mathbf{y}_h + \mathbf{a}_t \otimes \mathbf{y}_t - \mathbf{u}; \rho_1^{(2)}) = Com(\psi_h, \psi_t, \mathbf{a}_h \otimes \boldsymbol{v}_h + \mathbf{a}_t \otimes \mathbf{z}_t; \rho_1^{(3)}), \tag{9.1}$$

$$Com(\mathbf{x}_h, \mathbf{x}_t; \rho_2^{(1)}) = Com(\psi_h(\mathbf{z}_h), \psi_t(\mathbf{z}_t); \rho_2^{(3)}), \tag{9.2}$$

$$Com(\mathbf{w}_h + \mathbf{x}_h, \mathbf{w}_t + \mathbf{x}_t; \rho_3^{(1)}) = Com(\phi_h(\mathbf{y}_h), \phi_t(\mathbf{y}_t); \rho_3^{(2)}). \tag{9.3}$$

By the assumption that *Com* is computationally binding, there exists no distinct pair of arguments of *Com* in the equations (9.1), (9.2), and (9.3). From the equation (9.3), we have that $\mathbf{y}_h = \phi_h^{-1}(\mathbf{w}_h + \mathbf{x}_h) = \phi_h^{-1}(\mathbf{w}_h) + \phi_h^{-1}(\mathbf{x}_h)$ and $\mathbf{y}_t = \phi_t^{-1}(\mathbf{w}_t) + \phi_t^{-1}(\mathbf{x}_t)$. Combining the equations $\phi_h = \psi_h$, $\phi_t = \psi_t$, and (9.2), we obtain that $\mathbf{z}_h = \phi_h^{-1}(\mathbf{x}_h)$ and $\mathbf{z}_t = \phi_t^{-1}(\mathbf{x}_t)$. By substitution in the equation (9.1), we have that

$$\mathbf{a}_h \otimes (\phi_h^{-1}(\mathbf{w}_h) + \phi_h^{-1}(\mathbf{x}_h)) + \mathbf{a}_t \otimes (\phi_t^{-1}(\mathbf{w}_t) + \phi_t^{-1}(\mathbf{x}_t)) - \mathbf{u} = \mathbf{a}_h \otimes (\phi_h^{-1}(\mathbf{x}_h)) + \mathbf{a}_t \otimes (\phi_t^{-1}(\mathbf{x}_t)).$$

Simplifying the above we obtain that

$$\mathbf{a}_h \otimes (\phi_h^{-1}(\mathbf{w}_h)) + \mathbf{a}_t \otimes (\phi_t^{-1}(\mathbf{w}_t)) = \mathbf{u}.$$

Recall that $\mathcal{S}_h$ and $\mathcal{S}_t$ are enumeration sets. Therefore, $\phi_h^{-1}(\mathbf{w}_h) \in \mathcal{S}_h$ and $\phi_t^{-1}(\mathbf{w}_t) \in \mathcal{S}_t$. This completes the proof. □ □

9.4.4 Relations for NTRU

In this section, we tailor the relations for instantiations of NTRU.

Relations on Secret-Key Knowledge

For NTRU-1998: The secret keys $\mathbf{f}$ and $\mathbf{g}$ are chosen from $\mathcal{T}(d_f, d_f - 1)^*$ and $\mathcal{T}(d_g, d_g)$, respectively. Additionally, $\mathbf{f}$ must be invertible in $R_3 = \mathbb{Z}[\alpha]/(3, \alpha^n - 1)$. The public key is computed as $\mathbf{h} = 3\mathbf{g} \otimes \mathbf{f}^{-1}$.

We define the following relation:

$$\begin{aligned} R_{\text{KEY}}^{1998} = \{((-\mathbf{h}, 1, 0), (\mathbf{f}, 3\mathbf{g})) \in R_q^3 \times R_q^2 \\ : (-\mathbf{h} \otimes \mathbf{f} + 3\mathbf{g} = 0) \wedge (\mathbf{f} \in \mathcal{T}(d_f, d_f - 1)) \wedge (3\mathbf{g} \in 3\mathcal{T}(d_g, d_g))\}. \end{aligned}$$

For NTRU-2001: The secret key $\mathbf{f}$ is chosen from $\{1 + \mathbf{p} \otimes \mathbf{F} : \mathbf{F} \in \mathcal{B}(d_F)\}^*$, where $\mathbf{p} = 2 + \alpha$. The polynomial $\mathbf{g}$ is randomly chosen from $\mathcal{B}(d_g)$. The public key is computed as $\mathbf{h} = \mathbf{p} \otimes \mathbf{g} \otimes (1 + \mathbf{p} \otimes \mathbf{F})^{-1}$. We define the following relation:

$$\begin{aligned} R_{\text{KEY}}^{2001} := \{((-\mathbf{h}, 1, \mathbf{p}^{-1} \otimes \mathbf{h}), (\mathbf{F}, \mathbf{g})) \in R_q^3 \times R_q^2 \\ : (-\mathbf{h} \otimes \mathbf{F} + \mathbf{g} = \mathbf{p}^{-1} \otimes \mathbf{h}) \wedge (\mathbf{F} \in \mathcal{B}(d_F)) \wedge (\mathbf{g} \in \mathcal{B}(d_g))\}. \end{aligned}$$

For NTRU-2005: As in the relation on plaintext knowledge, we cannot define the relation on secret-key knowledge of the encryption in NTRU-2005, since $\mathcal{L}_F = \mathcal{X}(d_F)$ is not an enumeration set. Again, a simple modification allows us to define the key-pair relation for the modified version of NTRU-2005.

For NTRU-2007: The secret key $\mathbf{f}$ is chosen from $\{1 + 3\mathbf{F} : \mathbf{F} \in \mathcal{T}(d_f, d_f - 1)\}^*$. The polynomial $\mathbf{g}$ is randomly chosen from $\mathcal{T}(d_f, d_f - 1)^*$. The public key is computed as $\mathbf{h} = 3\mathbf{g} \otimes (1 + 3\mathbf{F})^{-1}$. Let 3^{-1} be the multiplicative inverse of 3 in R_q. We define the following relation:

$$\begin{aligned} R^{2007}_{\text{KEY}} := \{((-\mathbf{h}, 1, 3^{-1} \otimes \mathbf{h}), (\mathbf{F}, \mathbf{g})) \in R_q^3 \times R_q^2 \\ : (-\mathbf{h} \otimes \mathbf{F} + \mathbf{g} = 3^{-1} \otimes \mathbf{h}) \wedge (\mathbf{F} \in \mathcal{T}(d_f, d_f - 1)) \wedge (\mathbf{g} \in \mathcal{T}(d_f, d_f - 1))\}. \end{aligned}$$

For NTRU-2008: The secret key $\mathbf{f}$ is chosen from $\{1 + 3\mathbf{F} : \mathbf{F} \in \mathcal{T}(d_F, d_F)\}^*$. The polynomial $\mathbf{g}$ is randomly chosen from $\mathcal{T}(d_g, d_g)$. The public key is computed as $\mathbf{h} = 3\mathbf{g} \otimes (1 + 3\mathbf{F})^{-1}$. We define the following relation:

$$\begin{aligned} R^{2008}_{\text{KEY}} = \{((-\mathbf{h}, 1, \mathbf{h}), (3\mathbf{F}, 3\mathbf{g})) \in R_q^3 \times R_q^2 \\ : (-\mathbf{h} \otimes 3\mathbf{F} + 3\mathbf{g} = \mathbf{h}) \wedge (3\mathbf{F} \in 3\mathcal{T}(d_F, d_F)) \wedge (3\mathbf{g} \in 3\mathcal{T}(d_g, d_g))\}. \end{aligned}$$

Remark 9.4.3. We note that these relations do not imply that $\mathbf{f}$ and $\mathbf{g}$ are invertible in R_q. Moreover, these relation for NTRU-1998 does not assure that $\mathbf{f}$ is invertible in R_p. They guarantee that the l_∞ norm of $\mathbf{f}$ and $\mathbf{g}$ is relatively short and one can decrypt ciphertexts by using the polynomials $\mathbf{f}$ and $\mathbf{g}$ in the instantiations except NTRU-1998. In NTRU-1998, the keys satisfying the relation would imply the partial decryption.

Relations on Plaintext Knowledge

Recall the encryption procedure of NTRU. Let $\mathbf{m}$ and $\mathbf{r}$ be a plaintext and a randomness, respectively. The ciphertext $\mathbf{c}$ is $\mathbf{m}+\mathbf{h}\otimes\mathbf{r}$. Instead of $\mathcal{T}$, we use $\mathcal{T}(d_m, d_m)$ for some d_m. By changing the message spaces, each $\mathcal{L}_m$ is treated as an enumeration set.

For NTRU-1998 and NTRU-2008: $\mathcal{L}_r$ is set as $\mathcal{T}(d_r, d_r)$. We define the following relation:

$$\begin{aligned} R^{1998}_{\text{ENC}} = R^{2008}_{\text{ENC}} = \{((\mathbf{h}, 1, \mathbf{c}), (\mathbf{r}, \mathbf{m})) \in R_q^3 \times R_q^2 \\ : (\mathbf{h} \otimes \mathbf{r} + \mathbf{m} = \mathbf{c}) \wedge (\mathbf{r} \in \mathcal{T}(d_r, d_r)) \wedge (\mathbf{m} \in \mathcal{L}_m)\}. \end{aligned}$$

For NTRU-2001: In this case, $\mathcal{L}_r$ is set as $\mathcal{B}(d_r)$. We define the following relation:

$$R^{2001}_{\text{ENC}} := \{((\mathbf{h}, 1, \mathbf{c}), (\mathbf{r}, \mathbf{m})) \in R_q^3 \times R_q^2 : (\mathbf{h} \otimes \mathbf{r} + \mathbf{m} = \mathbf{c}) \wedge (\mathbf{r} \in \mathcal{B}(d_r)) \wedge (\mathbf{m} \in \mathcal{L}_m)\}.$$

For NTRU-2005: $\mathcal{X}(d_r)$ is defined as $\{\mathbf{f}_1 \otimes \mathbf{f}_2 + \mathbf{f}_3 : \mathbf{f}_i \in \mathcal{B}(d_r)\}$ and is not an enumeration set. We cannot define the relation on plaintext knowledge of the encryption in NTRU-2005, since $\mathcal{X}(d_r)$ is not an enumeration set.

However, simple modification allows us to define the relation for NTRU-2005 as in the above. For example, we could use $\mathcal{X}'(d_r) = \{\mathbf{f} \in \mathcal{X}(d_r) : \mathbf{f} \in \{0,1\}^n\}$, which is an enumeration set, instead of $\mathcal{X}(d_r)$.

For NTRU-2007: $\mathcal{L}_r$ is set as $\mathcal{T}(d_f, d_f - 1)$ and so is $\mathcal{L}_m$. We define the following relation:

$$\begin{aligned} R_{\mathrm{ENC}}^{2007} := \{((\mathbf{h}, 1, \mathbf{c}), (\mathbf{r}, \mathbf{m})) \in R_q^3 \times R_q^2 \\ : (\mathbf{h} \otimes \mathbf{r} + \mathbf{m} = \mathbf{c}) \wedge (\mathbf{r} \in \mathcal{T}(d_f, d_f - 1)) \wedge (\mathbf{m} \in \mathcal{T}(d_f, d_f - 1))\}. \end{aligned}$$

Remark 9.4.4. In order to prevent information leakage on **m**, NTRU-2008 recommended that the numbers of 1s, −1s, and 0s in a plaintext are at least some parameter. (See [WHGH$^+$08, HHHGW09, Section 9.2.2]).

Additionally, we note that certain encryption schemes used the enumeration set $\mathcal{B}(d)$ as the plaintext spaces. For example, the Chor–Rivest cryptosystem and the Okamoto–Tanaka–Uchiyama cryptosystem did so.

9.5 Identification Schemes

We can simply develop identification schemes based on NTRU from the XT protocol; A key-generation algorithm is same as the one in NTRU. A prover and a verifier runs the protocol for secret-key knowledge t times sequentially or in parallel. Let us discuss the security of this identification scheme. In the following, the security parameter n specifies the parameters N, p, and q, and the spaces $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_m$, and $\mathcal{L}_r$.

Assumptions: In the literature of padding schemes for NTRU, their securities are build on the one-way or the partial one-way assumption; the standard one-way assumption is stated as follows:

Definition 9.5.1 (The NTRU (one-way) assumption)**.** It is asymptotically hard to solve the NTRU inversion problem; For any polynomial-time adversary $\mathcal{A}$, the success probability $\mathbf{Adv}_{\mathrm{NTRU},\mathcal{A}}^{\mathrm{ow}}(n)$ is negligible in n; where

$$\mathbf{Adv}_{\mathrm{NTRU},\mathcal{A}}^{\mathrm{ow}}(n) = \Pr\left[\begin{array}{l} \mathcal{A}(\mathbf{h}, \mathbf{c}) = \mathbf{m} : \\ (\mathbf{h}, \mathbf{f}) \leftarrow \mathsf{KG}(1^n); \mathbf{m} \leftarrow \mathcal{L}_m; \mathbf{r} \leftarrow \mathcal{L}_r; \mathbf{c} = \mathbf{h} \otimes \mathbf{r} + \mathbf{m} \end{array}\right].$$

The problem on recovering the secret key is not easier than the problem on inverting the NTRU function, since, if one can get the secret key, then one can decrypt the ciphertexts.

Definition 9.5.2 (The NTRU factoring assumption)**.** This assumption states that it is asymptotically hard to solve the NTRU factoring problem; For any polynomial-time adversary $\mathcal{A}$, the success probability $\mathbf{Adv}^{\mathrm{fac}}_{\mathrm{NTRU},\mathcal{A}}(n)$ is negligible in n; where

$$\mathbf{Adv}^{\mathrm{fac}}_{\mathrm{NTRU},\mathcal{A}}(n) = \Pr\left[\begin{array}{l}(\mathbf{h} = p \otimes \mathbf{f}'^{-1} \otimes \mathbf{g}') \wedge (\mathbf{f}' \in \mathcal{L}_f) \wedge (\mathbf{g}' \in \mathcal{L}_g) : \\ (\mathbf{h}, \mathbf{f}) \leftarrow \mathsf{KG}(1^n); (\mathbf{f}', \mathbf{g}') \leftarrow \mathcal{A}(\mathbf{h})\end{array}\right].$$

Note that in the NTRU factoring assumption, the adversary has to output $\mathbf{f}'$ which has an inverse in R_q.

As stated in Remark 9.4.3, the XT protocol does not ensure that the prover has $\mathbf{f}$ which is invertible in R_q. Thus, we need another assumption which may be stronger then the NTRU factoring assumption.

Definition 9.5.3 (The NTRU decomposotion assumption)**.** This assumption states that it is asymptotically hard to solve the NTRU decomposition problem; For any polynomial-time adversary $\mathcal{A}$, the success probability $\mathbf{Adv}^{\mathrm{dec}}_{\mathrm{NTRU},\mathcal{A}}(n)$ is negligible in n; where

$$\mathbf{Adv}^{\mathrm{dec}}_{\mathrm{NTRU},\mathcal{A}}(n) = \Pr\left[\begin{array}{l}(\mathbf{f}' \otimes \mathbf{h} = p \otimes \mathbf{g}') \wedge (\mathbf{f}' \in \mathcal{L}_f) \wedge (\mathbf{g}' \in \mathcal{L}_g) : \\ (\mathbf{h}, \mathbf{f}) \leftarrow \mathsf{KG}(1^n); (\mathbf{f}', \mathbf{g}') \leftarrow \mathcal{A}(\mathbf{h})\end{array}\right].$$

The adversary violating this assumption still can be used to invert the NTRU function: Invoking the adversary, we obtain $\mathbf{f}'$ and $\mathbf{g}'$ such that $\mathbf{f}' \in \mathcal{L}_f$, $\mathbf{g}' \in \mathcal{L}_g$, and $\mathbf{f}' \otimes \mathbf{h} = p \otimes \mathbf{g}'$. Assume that the ciphertext is in the form $\mathbf{c} = \mathbf{h} \otimes \mathbf{r} + \mathbf{m}$. Multiplying $\mathbf{f}'$ to the ciphertext, We have that $\mathbf{f}' \otimes \mathbf{c} = p \otimes \mathbf{g} \otimes \mathbf{r} + \mathbf{f}' \otimes \mathbf{m}$ in R_q. Since the parameters are set to decrypt correctly with overwhelming probability, we can compute $\mathbf{a}' = p \otimes \mathbf{g} \otimes \mathbf{r} + \mathbf{f}' \otimes \mathbf{m}$ over $\mathbb{Z}$. Hence, we obtain $\mathbf{f}' \otimes \mathbf{m}$ in R_p. In the case of NTRU-2008, $\mathcal{L}_f$ is $\{1 + p\mathbf{F}\}$. Hence, we can correctly compute $\mathbf{m}$. In the case of NTRU-1998, $\mathcal{L}_f$ is $\mathcal{T}(d_f, d_f - 1)$. Even if $\mathbf{f}'$ is not invertible in R_p, we can partially decrypt $\mathbf{m}$ as stated in Remark 9.4.3. Consequently, the NTRU decomposition assumption is not stronger than the NTRU one-way assumption.

9.5.1 Description

As stated in the first paragraph of this section, we can develop a passive-secure identification scheme based on NTRU from the XT protocol for secret-key knowledge, since the protocol composed sequentially is a proof of knowledge and statistical zero knowledge.

Let *Com* be the special type of a statistically-hiding and computationally-binding string-commitment scheme as in Section 5.1.3.

Scheme 9.5.4 ($\mathsf{NTRU\text{-}ID}$)**.**

$\mathsf{Setup}(1^n)$**:** Given 1^n, output 1^n.

$\mathsf{KeyGen}(1^n)$**:** Choose $\mathbf{f} \leftarrow \mathcal{L}_f$ and $\mathbf{g} \leftarrow \mathcal{L}_g$ with the constrain that $\mathbf{f}$ is invertible in R_q and R_p. Set $\mathbf{F}_q \leftarrow \mathbf{f}^{-1}$ in R_q. Compute $\mathbf{h} \leftarrow p \otimes \mathbf{g} \otimes \mathbf{F}_q$ in R_q. The public key is $\mathbf{h}$ and the secret key is $(\mathbf{f}, \mathbf{g})$.

P **and** V**:** The common input is the pubic key $\mathbf{h} \in R_q$. Prover's auxiliary input is $(\mathbf{f}, \mathbf{g})$. The prover P and the verifier V interact as follows: The prover and the verifier runs the XT protocol for R_{KEY} in t-parallel or t-sequential.

9.5.2 Security Proofs

Theorem 9.5.5. *Let* NTRU-ID$_s$ *and* NTRU-ID$_p$ *denote obtained ID schemes by the sequential composition and by the parallel composition t times, respectively. Assume that there exists an adversary $\mathcal{A}$ that impersonates the valid prover with probability at least $\mathbf{Adv}^{\mathrm{IMP-PA}}_{\mathrm{ID},\mathcal{A}}(n)$. Then, there exists an adversary $\mathcal{B}$ which solves the NTRU decomposition problem with probability at least $\mathbf{Adv}^{\mathrm{dec}}_{\mathrm{NTRU},\mathcal{B}}(n)$ or breaks the binding property of the commitment scheme with probability at least $\mathbf{Adv}_{\mathrm{COM},\mathcal{B}}(n)$.*

The proof is obtained by applying the argument of Poupard and Pointcheval [PP03] to the proof by Stern [Ste96].

Proof. We only give the proof for the sequential composition, since the proof for parallel composition is very similar to the one in Section 6.7. We note that the proof for the sequential composition is also very similar to the ones of Stern [Ste96] and Pointcheval and Poupard [PP03].

Assume that there exists a polynomial-time adversary $\mathcal{A}$ that impersonates the prover with probability ϵ. We first replace the prover oracle with the simulation. This change introduces the statistical distance $Qt\Delta$.

Let ω denote the random tape of the adversary $\mathcal{A}$. Let I denote the random tape of the verifier that is identified with the challenge $C \in \{0, 1, 2\}^t$. Let us denote by S the set of the pairs (ω, I) which lead to acceptance. Hence, we have that $\Pr_{(\omega,I)}[(\omega, I) \in S] = \epsilon = (2/3)^t + \epsilon'$. Next, we define the set $\Omega = \{\omega \mid \Pr_I[(\omega, I) \in S] \geq (2/3)^t + \epsilon'/2\}$. A standard argument shows that $\Pr_\omega[\omega \in \Omega] \geq \epsilon'/2$ and $\Pr[\Omega \mid S] \geq \epsilon'/2\epsilon$. Assume in the following that the event Ω occurs.

Next, consider the execution tree $T(\omega)$, corresponding to all accepted I, with a fixed ω. We denote by n_i the number of the nodes at the depth i. We know that $n_0 = 1$ and $n_t = 2^t + 3^t\epsilon'/2$, because $n_k/3^k = \Pr_I[(\omega, I) \in S] \geq (2/3)^t + \epsilon'/2$. So, we have that

$$\prod_{i=0}^{t-1} \frac{n_{i+1}}{n_i} = \frac{n_t}{n_0} \geq 2^t + \frac{\epsilon'}{2} \cdot 3^t \geq \left(1 - \frac{\epsilon'}{2}\right) \cdot 2^t + \frac{\epsilon'}{2} \cdot 3^t.$$

By taking the logarithm of the inequation and using the convexity of the logarithm, we obtain that

$$\sum_{i=0}^{t-1} \log \frac{n_{i+1}}{n_i} \geq \left(1 - \frac{\epsilon'}{2}\right) \cdot \log 2^t + \frac{\epsilon'}{2} \cdot \log 3^t \geq t\left(\log 2 + \frac{\epsilon'}{2} \log \frac{3}{2}\right).$$

Therefore, there exists $i < t$ such that

$$\frac{n_{i+1}}{n_i} \geq 2(3/2)^{\epsilon'/2} = 2\exp\left(\frac{\epsilon'}{2} \cdot \log \frac{3}{2}\right) \geq 2 \cdot \left(1 + \frac{\epsilon'}{2} \cdot \log \frac{3}{2}\right) \geq 2 \cdot \left(1 + \frac{\epsilon'}{5}\right).$$

Let f_i and t_i denote the number of nodes at depth i with exactly 3 sons and that with at most 2 sons, respectively: We have that

$$n_i = f_i + t_i \text{ and } n_{i+1} \leq 3f_i + 2t_i = f_i + 2n_i.$$

Therefore, for the above i, we obtain that $2 + f_i/n_i \geq n_{i+1}/n_i 2 + 2\epsilon'/5$. Thus, so, with probability greater than $2\epsilon'/5$, the path I contains a node with 3 sons.

Now, the strategy of the reduction is as follows: (1) choose a random tape ω for the adversary $\mathcal{A}$. (2) choose a random challenge I for the simulated verifier. (3) using a ZK simulator, simulating the prover oracle. (4) checking $3k$ possible nodes along the path I. With probability greater than $\epsilon(\epsilon'/2\epsilon)(2\epsilon'/5) = \epsilon'^2/5$, we have found a node with 3 sons.

Assume that we have found a node with 3 sons. In that case, the reduction algorithm can obtains a collision for the commitment or solves the problem as in the proof of Stern [Ste96].

Hence, we have that

$$\frac{\epsilon'^2}{5} - Qt\Delta \leq \mathbf{Adv}^{\mathrm{dec}}_{\mathrm{NTRU},\mathcal{B}}(k) + \mathbf{Adv}_{\mathcal{COM},\mathcal{B}}(k).$$

□

Remark 9.5.6. Since the protocol is (cheating-verifier) zero knowledge, the reduction algorithm can simulate the valid prover even if the adversary accesses the prover oracle in an active way. This reduction requires as many steps as Qtk times of the original reduction. For simplicity, we only consider the reduction for the passive adversary.

9.5.3 Parameters and Communication Costs

In order to achieve the 80-bit security, we can set

$$(2/3)^t \leq 2^{-81}, \text{ and } \mathbf{Adv}^{\mathrm{dec}}_{\mathrm{NTRU},\mathcal{B}}(k), \mathbf{Adv}_{\mathrm{COM},\mathcal{B}}(k), Qt\Delta \leq 2^{-166}.$$

By solving $(2/3)^t \leq 2^{-81}$, we have that $t \geq 138.47...$ and set $t = 150$ (with reasonable margin). We use NTRU and the hash function which is suitable for use at the 192-bit security level. By setting $Q = 2^{60}$, we have $\Delta \leq 2^{-234.8137...}$ and set $\Delta = 2^{-256}$.

In NTRU-2008 [WHGH$^+$08, HHHGW09], three parameter sets, ees677ep1, ees887ep1, and ees1087ep1, are recommended for the 192-bit security level. We adopt ees677ep1: the public-key length is $677 \cdot \log 2048 = 7447$ bits and the secret-key length is $677 \cdot 2 = 1354$ bits.

We next adopt the Halevi-Micali commitment scheme [HM96]. In this case, we need a 192-bit secure cryptographic hash function which outputs the digest of length at least 384 bits. Then, we have the commitment scheme where the length of the commitment is $7 \cdot 384 = 2688$ bits and the length of the decommitment is

$|m| + 2688$ bits (m is a message to be committed). We have to compute $|m|$. The prover will sends two permutations and two vectors, or four vectors in Step P2. In the former, the length of the message is $2 \cdot \log 677! + 2 \cdot 667 \cdot \log 2048 \sim 25686$ bits. In the latter, the length of the message is $2 \cdot 677 \cdot \log 3 + 2 \cdot 677 \cdot \log 2048 \sim 17041$ bits.

Thus, the total communication cost is $150 \cdot (3 \cdot 2688 + 2 + (25686 + 2 \cdot 2688)) = 5869200$ bits (approximately 716.5kB).

Discussions: By parallel composition, we can obtain almost the same identification scheme. However, this scheme has a drawback in the sense of tightness of the reduction. In this case, the tightness is cubic as in Kawachi et al. [KTX08], rather than quadratic as in the above.

It seems that concurrently secure identification schemes need stronger assumptions, such as the one-more NTRU one-way assumption or the assumption that the small integer solution problem over NTRU lattices is hard on the average. The small integer solution problem, SIS_β, is, given a matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$, to find a non-zero vector $z \in \Lambda_q^\perp(\boldsymbol{A})$ such that $\|z\| \leq \beta$ in some norm. Using this assumptions, Lyubashevsky [Lyu08a] and Kawachi et al. [KTX08] succeeded to construct concurrently secure identification schemes based on lattice problems.

9.6 Comparisons

There are several identification schemes based on combinatorial problems. We compare the schemes such as Stern's SD-based [Ste96], Shamir's PKP-based [Sha89], PPP-based by Pointcheval and Poupard [PP03], Lyubashevsky's C/IL-based [Lyu08a, Lyu08b], and C/IL-based by Kawachi et al. [KTX08] identification schemes. For the comparison with standard identification schemes, we put GQ [GQ88] and Schnorr [Sch91] in Table 9.2.

In the papers [Ste96, Sha89], the authors ignored the commitment scheme and directly used the hash value. Thus, the proofs are not correct in the standard model (this requires stronger assumptions on the hash function). We, hence, replace the hash function with the commitment scheme.

The main difficulty of comparison is that the parameter settings for other schemes were not explicit. They did not propose the parameter-generating method in order to attain the security level. Here, we briefly discuss the parameters and costs if we set 80-*bit security* for the identification schemes.

Storage costs: Notice that, in all reduction, the advantages of the adversary against identification schemes are upperbounded by the square roots of the advantage of the adversary against the assumption that the underlying problem is hard. Thus, the best work factor for solving the underlying problem must be at least 2^{160}. We here require 196-bit security for the underlying problem.

Recall that in the case of NTRU-ID, we have a 7447-bit public key.

Name	Reduction	Security	$\lvert pk \rvert$ (kB)	Cost (kB)	Ref.
NTRU-ID	$(2/3)^t + \sqrt{5}\sqrt{\epsilon(k) + \epsilon'(k) + Qt\Delta}$	P (A)	0.91	716.5	[XT09]
St-ID$_{\mathsf{p}}$	$(2/3)^t + \sqrt{5}\sqrt{\epsilon(k) + \epsilon'(k) + Qt\Delta}$	P (A)	0.204 (+680.9)	935.1	[Ste96]
PKP	$\left(\frac{126}{251}\right)^t + \sqrt{5.758...}\sqrt{\epsilon(k) + \epsilon'(k) + Qt\Delta}$	P (A)	≥ 2.313	≥ 90.0	[Sha89]
PPP	$(3/4)^t + \sqrt{14/3}\sqrt{\epsilon(k) + \epsilon'(k) + 2Qt\Delta}$	P (A)	$\geq$0.196 (+$\geq$5.324)	≥ 637.0	[PP03]
Ly09-ID	$\sqrt{2\epsilon(k) + 4 \cdot 2^{-kt/5} + 2^{-3k \log k/4}}$	C	?	?	[Lyu09]
St-ID$^{+}_{\mathsf{C/IL,p}}$	$(2/3)^t + \sqrt{10}\sqrt{2\epsilon(k) + Qt\Delta'} + 2^{-\Omega(k)}$	C	?	?	[KTX08]
(cf:) GQ	$(1/2)^{l(k)} + \sqrt{\epsilon(k)}$	P	0.938 (+1.875)	1.885	[GQ88]
(cf:) Schnorr	$(1/2)^{l(k)} + \sqrt{\epsilon(k)}$	P	0.938 (+0.957)	0.967	[Sch91]

Table 9.2: Comparisons on reduction and security. Note: NTRU-ID, St-ID$_{\mathsf{p}}$, PKP, PPP, and St-ID$^{+}_{\mathsf{C/IL,p}}$ run the basic protocol in t times sequentially. In Ly09-ID, t is the number for parallel. Each $\epsilon(k)$ denotes the advantage of the polynomial-time adversary against the underlying problem. Each $\epsilon'(k)$ denotes $\mathbf{Adv}^{\mathrm{bd}}_{\mathrm{COM}}(k)$, the advantage of the polynomial-time adversary against the commitment scheme. In St-ID$^{+}_{\mathsf{C/IL,p}}$, Δ' denotes the regularity of the lattice-based hash functions. In GQ and Schnorr, $l(k)$ denotes the length of the challenge message. In the column of security, P, A, and C denotes passive, active, and concurrent, respectively.

In the GQ and Schnorr schemes, the RSA modulus N or the parameter for the group p is often of length 2048 bits. However, to achieve 196-bit security, they should be 7680-bit numbers. Interestingly, in the GQ scheme, the public key consists of N and two elements e, X in $\mathbb{Z}_N^*$, so, the length of public key is longer than that of NTRU-ID. Even if (N, e) in the GQ scheme and p in the Schnorr scheme is public parameter, there is a little difference between the length of the public key.

In Stern's scheme St-ID$_{\mathsf{p}}$ [Ste96], in order to achieve 196-bit security, it requires a random matrix in $\mathbb{Z}_q^{n \times m}$ as a public parameter, where $(q, n, m) = (2, 1670, 3340)$ (see the estimation by Canteaut and Chabaud [CC98, Approximation 1]). Hence it requires approximately 680kB for the public parameter. Each public key is a vector in $\mathbb{Z}_q^n$, whose length is 1670 bits.

In the PKP scheme, Shamir proposed the parameter set $(q, n, m) = (251, 16, 32)$ and $(q, n, m) = (251, 37, 64)$, which may achieve 76-bit and 184-bit security. We adopt the latter parameter set. In the case, the public key $\boldsymbol{A}$ is a matrix in $\mathbb{Z}_q^{n \times m}$ and the length of it is 18944 bits (approximately 2.31kB).

In the PPP scheme by Poupard and Pointcheval [PP03], they proposed several parameters $(n, m) = (121, 137)$ and $(n, m) = (201, 217)$, which are estimated 2^{64}-security and more. Hence, we adopt the latter. Note that their public parameters are of length at least $201 \cdot 217 = 43617$ bits. Each public key is a vector in $\mathbb{Z}_m^n$, whose length is approximately 1560 bits.

Communication Costs: The communication costs mainly depend on the domain size of the permutations in the protocol. In PKP and PPP, they used a permutation over [64] or over [217], respectively. Thus, their communication costs are relatively low. On the other hand, $\mathsf{St\text{-}ID_p}$ uses a permutation over [3340] and $\mathsf{NTRU\text{-}ID}$ employs two permutations over [677]. Hence, the communication costs are very large.

9.7 Concluding Remarks

On computational-zero-knowledge proof systems: By using the hard-core bit of NTRU [NSW03] or the general hard-core predicate by Goldreich and Levin, we can construct computationally-hiding and statistically-binding bit-commitment schemes from the NTRU one-way assumption. By using the above commitment scheme, one obtains computational-zero-knowledge and proof-of-knowledge proof systems for the same relations.

Signature schemes: In the literature, there were two practical signature schemes based on NTRU, NSS [HPS01] and NTRUSign [HHGP$^+$03]. Unfortunately, their security were not proven under plausible assumptions. Indeed, NSS and a simple version of NTRUSign were already broken [GJSS01, NR06].

Meanwhile, applying the Fiat-Shamir transformation to the 3-round parallelized version of the XT protocol, we can obtain a secure signature scheme under the NTRU decomposition assumption in the random oracle model. However, the XT-NTRU signature scheme is far from practical use.

We finally mention the signature scheme by Gentry, Peikert, and Vaikuntanathan [GPV08]. In their scheme, the public key is $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and the secret key is a short basis of $\Lambda_q^{\perp}(\boldsymbol{A})$ in the l_2 norm. We already saw that the short vector $\mathbf{f} \circ (p\mathbf{g})$ is in $\Lambda_q^{\perp}([-\operatorname{Rot}(\mathbf{h})\ \operatorname{Rot}(1)])$. By rotating the short vector, one can obtain a half of the basis of the NTRU lattice. Hoffstein, Howgrave-Graham, Pipher, Silverman, and Whyte proposed the NTRUSign [HHGP$^+$03] in 2003. In [HHGP$^+$03], they discussed how to obtain the remaining half of the basis of the NTRU lattice. They used certain norm rather than the l_2 norm. The method obtaining the remaining half of a short basis in l_2 norm would yield a secure signature scheme based on the NTRU problems in a similar way to the GPV signature scheme [GPV08].

10
Trapdoors for Lattices

Organization: We briefly introduce the background of trapdoor functions based on lattice problems in Section 10.1. In Section 10.2, we review the definition of (one-way and collision-resistant) preimage sampleable functions (PSFs), which suit for lattice-based trapdoor functions. In Section 10.3, we review the Alwen-Peikert algorithm for trapdoor generation. Section 10.4 reviews the sampling algorithm for $D_{\Lambda,s,\boldsymbol{c}}$ by Gentry, Peikert, and Vaikuntanathan (which appears originally in Klein [Kle00]). In Section 10.5, we describes lattice-based PSFs obtained by combining the aboves. Section 10.6 illustrates the ideal version of the Alwen-Peikert construction. In Section 10.7 describes an instantiation of PSFs from ideal lattices. Section 10.8 reviews the notions of "Bonsai" techniques. We apply these techniques to ideal-lattice-based constructions in Section 10.9. As direct applications of PSFs, we construct lattice-based trapdoor hash functions in Section 10.10

10.1 Introduction

In the seminal paper of Ajtai [Ajt96], he gave an instance-generation algorithm for SIS that outputs $(\boldsymbol{A}, \boldsymbol{e})$: Generate a random vector $\boldsymbol{e} \leftarrow \{0, 1\}^m$, generates a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ with constrain that $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{0}$, and permutes them. But, an instance-generation algorithm that outputs $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n \times m}$ with the short basis of $\Lambda_q^{\perp}(\boldsymbol{A})$ is non-trivial. After this algorithm, he proposed the instance generation algorithm for this problem [Ajt99]. This algorithm was an isolated point of lattice-based cryptography; because in about decade, there were no cryptographic schemes employing this algorithm.

In 2008, Gentry, Peikert, and Vaikuntanathan [GPV08] showed that the short basis has a power of sampling the discretized Gaussian $D_{\Lambda,s,\boldsymbol{c}}$ on the lattice Λ. They also improved the analysis of the Ajtai algorithm. Alwen and Peikert further improved the Ajtai algorithm [AP09]. Finally, in 2009, Stehlé, Steinfeld, Tanaka, and Xagawa [SSTX09] proposed an ideal version of the Alwen–Peikert construction.

These algorithm allows to implement trapdoors for $h_{\boldsymbol{A}}$ (or $h_{\check{\boldsymbol{a}}}$). The trapdoor is a basis $\boldsymbol{T}$ of a lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ (or $\Lambda_q^{\perp}(\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}}))$) such that $\|\tilde{\boldsymbol{T}}\| \leq L$ for some L. One can sample $D_{\Lambda_q^{\perp}(\boldsymbol{A}),s,\boldsymbol{c}}$ (or $D_{\Lambda_q^{\perp}(\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})),s,\boldsymbol{c}}$) by using $\boldsymbol{T}$ for $s = L \cdot \omega(\sqrt{\log n})$. Turning it into hash functions, one can sample preimages $\boldsymbol{e}$ (or $\check{\boldsymbol{e}}$) of $\boldsymbol{u}$ (or $\mathbf{u}$). Since appearing distributions are not uniform, Gentry et al. defined preimage sampleable functions rather than trapdoor functions for generality.

10.2 Definition of Preimage Sampleable Functions

Roughly speaking, preimage sampleable functions (PSFs) is a hash family $\mathcal{H} = \{\mathcal{H}_n\}$, where $\mathcal{H}_n = \{f_a : D_n \to R_n \mid (a,t) \in K_n \times T_n\}$, defined with a distribution ensemble $X = \{X_n\}$ over $D = \{D_n\}$. First, one can *sample* preimages of $y \in R_n$ under f_a by using the corresponding t to a. Next, the two distributions of the samples (x, y) and (x', y') must be statistically identical, where (x, y) is sampled by $x \leftarrow X_n$ and $y \leftarrow f_a(y)$ and (x', y') is sampled by $y' \leftarrow U(D_n)$ and obtaining x' by the above trapdoor sampling procedure. In addition, the distribution $f_a(X_n)$ is almost uniform over R_n.

Gentry et al. [GPV08] defined it in the algorithmic form and we follow them.

10.2.1 Model of Preimage Sampleable Functions

The preimage sampleable (trapdoor) functions PSF defined by a quadruplet of algorithms $(\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SampleDom}, \mathsf{SamplePre})$.

$\mathsf{TrapGen}(1^n)$: A trapdoor-generation algorithm, given the security parameter 1^n, outputs a description of function $a \in K_n$ and its trapdoor t. (Notice that a defines the function $f_a : D_n \to R_n$.)

$\mathsf{Eval}(a, x)$: An evaluation algorithm, given a and an element $x \in D_n$, returns $y = f_a(x)$.

$\mathsf{SampleDom}(1^n)$: A domain sampling algorithm, given the security parameter 1^n, samples $x \in D_n$ from some distribution over D_n.

$\mathsf{SamplePre}(t, y)$: A preimage sampling algorithm, given a trapdoor t corresponding to a and an image y, samples x from some distribution over D_n.

Definition 10.2.1 (Preimage Sampleable Functions)**.** We say PSF is preimage sampleable function scheme if the following conditions hold: Let X denote the random variable stands for the output of $\mathsf{SampleDom}$ and let X_y denote the random variable according to the conditional distribution of the output x by $\mathsf{SampleDom}$ given $f_a(x) = y$.

Domain sampling with uniform distribution: With overwhelming probability of the choice of a, $\mathsf{SampleDom}$ samples an x for which the distribution of $f_a(x)$ is statistically close to uniform over R_n. Formally,

$$\Pr\left[\Delta(f_a(X), U(R_n)) \leq \mathsf{negl}(n) : \quad (a,t) \leftarrow \mathsf{TrapGen}(1^n); \ \right] \geq 1 - \mathsf{negl}(n).$$

Preimage sampling with trapdoor: With overwhelming probability of the choice of a, $\mathsf{SamplePre}$, given t and y, samples an x for which the distribution of x is statistically close to that of X_y. Formally, for any $y \in R_n$,

$$\Pr\left[\Delta(\mathsf{SamplePre}(t,y), X_y) \leq \mathsf{negl}(n) : \quad (a,t) \leftarrow \mathsf{TrapGen}(1^n); \ \right] \geq 1 - \mathsf{negl}(n).$$

10.2.2 Security Notions

Roughly speaking, we say that PSF is one-way if any polynomial-time adversary cannot, given a and y, output a preimage x of y under f_a. We say that PSF is collision-resistant if any polynomial-time adversary cannot, given a, output distinct $x, x' \in D_n$ such that $f_a(x) = f_a(x')$ and the conditional min-entropy of $x \leftarrow \mathsf{SampleDom}(1^n)$ given $f_a(x) = y$ is at least $\omega(\log n)$. Note that the difference between the collision-resistance definitions of Hash and PSF. (The definition of the hash scheme does not require the min-entropy condition.)

Formally, we define the following experiments $\mathbf{Exp}^{\mathrm{ow}}_{\mathsf{PSF},\mathcal{A}}(n)$ and $\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{PSF},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\mathrm{ow}}_{\mathsf{PSF},\mathcal{A}}(n)$:

Setup Phase: The challenger C runs TrapGen with 1^n and obtains (a,t). Next, it generates $y \leftarrow R_n$ uniformly at random. C feeds a and y to the adversary $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ outputs x. If $x \in D_n$ and $f_a(x) = y$ then the challenger returns 1, otherwise, 0.

Experiment $\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{PSF},\mathcal{A}}(n)$:

Setup Phase: The challenger C runs TrapGen with 1^n and obtains (a,t). C feeds a to the adversary $\mathcal{A}$.

Challenge Phase: $\mathcal{A}$ outputs x and x' If $x, x' \in D_n$, $x \neq x'$, and $f_a(x) = f_a(x')$ then the challenger returns 1, otherwise, 0.

Definition 10.2.2. Let $\mathsf{PSF} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SampleDom}, \mathsf{SamplePre})$ be a preimage sampleable function scheme. Let $\mathcal{A}$ be an adversary. Let the advantage of $\mathcal{A}$ against one-wayness be $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{PSF},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{ow}}_{\mathsf{PSF},\mathcal{A}}(n) = 1\right]$. We say that PSF is one-way if, for any polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{ow}}_{\mathsf{Hash},\mathcal{A}}(n)$ is negligible in n.

Let the advantage of $\mathcal{A}$ against collision resistance be $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{PSF},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{PSF},\mathcal{A}}(n) = 1\right]$. We say that PSF is collision resistant if, for any polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{PSF},\mathcal{A}}(n)$ is negligible in n *and* the conditional min-entropy

$H_\infty(X \mid f_a(X) = y)$ is at least $\omega(\log n)$, where X denotes the random variable which stands for the output of $\mathsf{SampleDom}(1^n)$.

10.3 The Ajtai and Alwen–Peikert Constructions

We next review the one of the underlying component of lattice-based PSFs, the Ajtai and Alwen–Peikert constructions. As noted in Section 10.1, Ajtai [Ajt99] proposed that the instance-generation algorithm which outputs a random matrix $\boldsymbol{A}$ and a short basis $\boldsymbol{T}$ of a lattice $\Lambda_q^\perp(\boldsymbol{A})$. There are improved versions of the algorithm by Gentry et al. [GPV08] and by Alwen and Peikert [AP09]. We follow the construction by Alwen and Peikert.

10.3.1 Main Strategy

Assume that we have a random matrix $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n\times m_1}$. We then want to construct random $\boldsymbol{A}_2 \in \mathbb{Z}_q^{n\times m_2}$ with a short basis $\boldsymbol{S} \in \mathbb{Z}_q^{m\times m}$ of $\Lambda_q^\perp(\boldsymbol{A})$, where $m = m_1 + m_2$ and $\boldsymbol{A} = [\boldsymbol{A}_1|\boldsymbol{A}_2]$. Let $d = (1+\delta)n\log q$. We suppose that $m_1 \geq d$ which will support the uniformity of $\boldsymbol{A}_2$.

To construct $\boldsymbol{S}$, we first compute an Hermite normal form $\boldsymbol{H} \in \mathbb{Z}^{m_1\times m_1}$ of a basis of $\Lambda_q^\perp(\boldsymbol{A}_1)$. Since $\boldsymbol{H}$ is a basis of $\Lambda_q^\perp(\boldsymbol{A}_1)$, we have that $\boldsymbol{A}_1\boldsymbol{H} \equiv \boldsymbol{O} \pmod{q}$. With high probability, $\Lambda_q^\perp(\boldsymbol{A}_1)$ is full-rank, and so is $\boldsymbol{H}$.

Next, let us construct $\boldsymbol{F} = [\boldsymbol{H}|\boldsymbol{U}; \boldsymbol{O}|\boldsymbol{I}_{m_2}]$ for some $\boldsymbol{U} \in \mathbb{Z}^{m_1\times m_2}$ and $\boldsymbol{A}_2 \in \mathbb{Z}_q^{n\times m_2}$ such that $[\boldsymbol{A}_1|\boldsymbol{A}_2]\boldsymbol{F} \equiv \boldsymbol{O} \pmod{q}$. In order to do so, we set $\boldsymbol{A}_2 \equiv -\boldsymbol{A}_1\boldsymbol{U} \pmod{q}$ and we have $\boldsymbol{A}\boldsymbol{F} = [\boldsymbol{A}_1\boldsymbol{H}|\boldsymbol{A}_1\boldsymbol{U} + \boldsymbol{A}_2] \equiv \boldsymbol{O} \pmod{q}$, where $\boldsymbol{U}$ has randomness to applying the leftover hash lemma and will be defined later. Notice that $\boldsymbol{F}$ is a basis of $\Lambda_q^\perp(\boldsymbol{F})$ by construction.

We then construct a unimodular matrix $\boldsymbol{Q} = [-\boldsymbol{I}_{m_1}|\boldsymbol{O}; \boldsymbol{P}|\boldsymbol{B}]$ such that a basis $\boldsymbol{S} = \boldsymbol{F}\boldsymbol{Q}$ is short. We will set $\boldsymbol{B}$ an upper triangle matrix with diagonals 1, which yields the unimodularity of $\boldsymbol{Q}$. We figure them as follows:

$$\begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 \end{bmatrix}\begin{bmatrix} \boldsymbol{V} & \boldsymbol{D} \\ \boldsymbol{P} & \boldsymbol{B} \end{bmatrix} = \boldsymbol{O} \text{ and } \underbrace{\begin{bmatrix} \boldsymbol{V} & \boldsymbol{D} \\ \boldsymbol{P} & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{S}} = \underbrace{\begin{bmatrix} \boldsymbol{H} & \boldsymbol{U} \\ \boldsymbol{O} & \boldsymbol{I}_{m_2} \end{bmatrix}}_{\boldsymbol{F}}\underbrace{\begin{bmatrix} -\boldsymbol{I}_{m_1} & \boldsymbol{O} \\ \boldsymbol{P} & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{Q}}.$$

By setting $\boldsymbol{U} = \boldsymbol{R} + \boldsymbol{G}$, with $\boldsymbol{G}$ to be defined later on and $\boldsymbol{R}$ a random matrix, we will have that $\boldsymbol{A}_2$ is almost uniformly random by the leftover hash lemma. More precisely, we set $\boldsymbol{R} = [\boldsymbol{R}'; \boldsymbol{O}] \in \mathbb{Z}^{m_1\times m_2}$ and $\boldsymbol{R}'$ is chosen from $\{-1, 0, +1\}^{d\times m_2}$. According to the structure of $\boldsymbol{S}$, we have that

$$\boldsymbol{D} = (\boldsymbol{G} + \boldsymbol{R})\boldsymbol{B} \text{ and } \boldsymbol{V} = -\boldsymbol{H} + (\boldsymbol{G} + \boldsymbol{R})\boldsymbol{P}.$$

The matrix $\boldsymbol{G}$ will be designed to $\boldsymbol{G}\boldsymbol{P} = \boldsymbol{H}_2 - \boldsymbol{I}_d$. So, we let $\boldsymbol{V} = \boldsymbol{R}\boldsymbol{P} - \boldsymbol{I}_{m_1}$. Note that $\boldsymbol{D} = \boldsymbol{G}\boldsymbol{B} + \boldsymbol{R}\boldsymbol{B}$ and hence we let $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{B}$.

Preliminaries of the constructions: We first show that $\boldsymbol{F}$ is a basis of $\Lambda_q^{\perp}([\boldsymbol{A}_1|\boldsymbol{A}_2])$.

Lemma 10.3.1. *If $\boldsymbol{H}$ is a basis of $\Lambda_q^{\perp}(\boldsymbol{A}_1)$ then $\boldsymbol{F}$ is a basis of $\Lambda_q^{\perp}([\boldsymbol{A}_1|\boldsymbol{A}_2])$.*

Proof. Let $\boldsymbol{A} = [\boldsymbol{A}_1|\boldsymbol{A}_2]$. Consider any $\boldsymbol{e}_1 \circ \boldsymbol{e}_2 \in \Lambda_q^{\perp}(\boldsymbol{A})$. We have that $\boldsymbol{A} \cdot (\boldsymbol{e}_1 \circ \boldsymbol{e}_2) = \boldsymbol{A}_1\boldsymbol{e}_1 + \boldsymbol{A}_2\boldsymbol{e}_2 \equiv \boldsymbol{0} \pmod{q}$. Since $\boldsymbol{A}_2 = -\boldsymbol{A}_1\boldsymbol{U}$ in the construction, we also have that

$$\boldsymbol{A}_1(\boldsymbol{e}_1 - \boldsymbol{U}\boldsymbol{e}_2) \equiv \boldsymbol{0} \pmod{q}.$$

Thus, $\boldsymbol{e}_1 - \boldsymbol{U}\boldsymbol{e}_2 \in \Lambda_q^{\perp}(\boldsymbol{A}_1)$. This indicates there is some $\boldsymbol{w} \in \mathbb{Z}^{m_1}$ such that $\boldsymbol{H}\boldsymbol{w} = \boldsymbol{e}_1 - \boldsymbol{U}\boldsymbol{e}_2$, since $\boldsymbol{H}$ is a basis of $\Lambda_q^{\perp}(\boldsymbol{A})$.

We note that $\boldsymbol{F} \subseteq \Lambda_q^{\perp}(\boldsymbol{A})$ since $\boldsymbol{A}\boldsymbol{F} \equiv \boldsymbol{O} \pmod{q}$ by the construction. Since $q\boldsymbol{I}_{m_1} \subseteq \Lambda_q^{\perp}(\boldsymbol{A}_1)$, the basis $\boldsymbol{H}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_1)$ is full-rank. Thus, $\boldsymbol{F} = [\boldsymbol{H}|\boldsymbol{U};\boldsymbol{O}|\boldsymbol{I}_{m_2}]$ is also full-rank.

Hence, we can write $\boldsymbol{e}_1 \circ \boldsymbol{e}_2 = \boldsymbol{F}(\boldsymbol{c}_1 \circ \boldsymbol{c}_2)$ for some $\boldsymbol{c}_1 \in \mathbb{Q}^{m_1}$ and $\boldsymbol{c}_2 \in \mathbb{Q}^{m_2}$. It suffices to show they are integer vectors. This means $\boldsymbol{e}_1 = \boldsymbol{H}\boldsymbol{c}_1 + \boldsymbol{U}\boldsymbol{c}_2$ and $\boldsymbol{e}_2 = \boldsymbol{c}_2$. Thus, we have that $\boldsymbol{e}_1 = \boldsymbol{H}\boldsymbol{c}_1 + \boldsymbol{U}\boldsymbol{c}_2$ and $\boldsymbol{H}\boldsymbol{c}_1 = \boldsymbol{e}_1 - \boldsymbol{U}\boldsymbol{e}_2 \in \mathbb{Z}^{m_1}$. Hence, $\boldsymbol{c}_1 = \boldsymbol{w}$ and we have confirmed that $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ are integer vectors, which completes the proof. □

Notice that the determinant of $\boldsymbol{H}$ is at most q^n and each diagonal of $\boldsymbol{H}$ is at most q (the equality holds when the columns of $\boldsymbol{A}_1$ generates $\mathbb{Z}_q^n$).

Hereafter, we set $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{B}$. We often use the matrix $\boldsymbol{T}_\kappa = \{t_{i,j}\} \in \mathbb{Z}^{\kappa\times\kappa}$, where $t_{i,i} = 1$, $t_{i,i+1} = -r$, and all other $t_{i,j}$'s are 0. Illustratively,

$$\boldsymbol{T}_\kappa = \begin{array}{c|ccccc|} & 1 & 2 & 3 & \cdots & \kappa \\ 1 & 1 & -r & & & \\ 2 & & 1 & -r & & \\ 3 & & & 1 & \ddots & \\ \vdots & & & & \ddots & -r \\ \kappa & & & & & 1 \end{array}, \quad \boldsymbol{T}_\kappa^{-1} = \begin{array}{c|ccccc|} & 1 & 2 & 3 & \cdots & \kappa \\ 1 & 1 & r & r^2 & \dots & r^{\kappa-1} \\ 2 & & 1 & r & \dots & r^{\kappa-2} \\ 3 & & & 1 & \ddots & \vdots \\ \vdots & & & & \ddots & r \\ \kappa & & & & & 1 \end{array}.$$

It is easy to verify $\boldsymbol{T}_\kappa^{-1}$ is the inverse of $\boldsymbol{T}_\kappa$ by a multiplication.

There are three versions of the Alwen–Peikert construction. See the following sections (Section 10.3.2, Section 10.3.3, and Section 10.3.4).

10.3.2 The First Construction

Theorem 10.3.2 (Alwen and Peikert [AP09])**.** *Let $\delta > 0$ and $r \geq 2$ be any constant. Let $m_1 = m_1(n)$, $m_2 = m_2(n)$, $m = m_1 + m_2$, and $q = q(n)$. There is a probabilistic polynomial-time algorithm* ExtLattice1 *that, on input 1^n and uniformly random matrix $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n\times m_1}$, outputs a pair $(\boldsymbol{A} = [\boldsymbol{A}_1|\boldsymbol{A}_2], \boldsymbol{S}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}^{m\times m}$. If $m_1 \geq d = (1+\delta)n\log q$ and $m_2 \geq 2n\log q$,*

- *$\boldsymbol{A}$ is $(m_2 \cdot q^{-\delta n/2})$-uniform over $\mathbb{Z}_q^{n\times m}$,*

- *$\boldsymbol{S}$ is a basis of $\Lambda^{\perp}(\boldsymbol{A})$, and*
- *for any $\omega(\sqrt{\log n})$ functions, $\|\boldsymbol{S}\| \le (m_1 + n\log_r q)\cdot\omega(\sqrt{\log n})$ with overwhelming probability.*

Description: We start with a construction of $\boldsymbol{B}$; Let $\boldsymbol{H}' = \boldsymbol{H} - \boldsymbol{I}_{m_1}$. Let c_i and c'_i denote i-th diagonals of $\boldsymbol{H}$ and $\boldsymbol{H}'$, respectively. Notice that $c_i \in [1, q]$. Let $l_i = \lceil \log_r c_i \rceil \le 1 + \log_r c_i$. Define the partial sums $s_0 = 0$, $s_j = s_{j-1} + l_j$ for $j \in [m_1]$. Define the total sum $s = s_{m_1}$.

Note that if $c_i = 1$ then $l_i = 0$ and there are at most $n \log q$ values of i for which $c_i > 1$. In addition, we have that $\prod_{i\in[m_1]} c_i \le q^n$, since $\det(\boldsymbol{H}) \le q^n$. Therefore, we have that $s \le n\log q + \sum_i \log_r c_i \le 2n \log q \le m_2$.

Here, we set

$$\boldsymbol{B} = \mathrm{diag}(\boldsymbol{T}_{l_1}, \ldots, \boldsymbol{T}_{l_{m_1}}, \boldsymbol{I}_{m_2-s}).$$

We note that

$$\boldsymbol{B}^{-1} = \mathrm{diag}(\boldsymbol{T}_{l_1}^{-1}, \ldots, \boldsymbol{T}_{l_{m_1}}^{-1}, \boldsymbol{I}_{m_2-s}).$$

We next split $\boldsymbol{W}$ and $\boldsymbol{G}$ into $m_1 + 1$ matrices where $\boldsymbol{W} = [\boldsymbol{W}^{(1)}|\ldots|\boldsymbol{W}^{(m_1)}|\boldsymbol{O}]$, $\boldsymbol{G} = [\boldsymbol{G}^{(1)}|\ldots|\boldsymbol{G}^{(m_1)}|\boldsymbol{O}]$, and $\boldsymbol{W}^{(k)}, \boldsymbol{G}^{(k)} \in \mathbb{Z}^{m_1\times l_k}$ for any $k \in [m_1]$. Let $\boldsymbol{W}^{(k)} = \{w^{(k)}_{i,j}\}_{i\in[m_1], j\in[l_k]}$ and $\boldsymbol{G}^{(k)} = \{g^{(k)}_{i,j}\}_{i\in[m_1], j\in[l_k]}$. We set

$$w^{(k)}_{i,j} = \begin{cases} 1 & (i = k \text{ and } j = 1), \\ 0 & (\text{otherwise}) \end{cases}.$$

By this construction, we have that

$$g^{(k)}_{i,j} = \begin{cases} r^j & (i = k), \\ 0 & (\text{otherwise}) \end{cases}$$

since $\boldsymbol{G} = \boldsymbol{W}\boldsymbol{B}^{-1}$. Let $\boldsymbol{g}_l = [1, r, \ldots, r^{l-1}] \in \mathbb{Z}^{1\times l}$. Illustratively, we have that

$$\boldsymbol{G} = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ m_1 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & \cdots & m_1 & \cdots \end{array} \\ \left[\begin{array}{ccccc} \boldsymbol{g}_{l_1} & & & & \\ & \boldsymbol{g}_{l_2} & & & \\ & & \ddots & & \\ & & & \boldsymbol{g}_{l_{m_1}} & \end{array}\right] \end{array}$$

Using this construction, making $\boldsymbol{G}\boldsymbol{P} = \boldsymbol{H}' = \boldsymbol{H} - \boldsymbol{I}_{m_1}$ is straightforward; Let $\boldsymbol{P} = [\boldsymbol{P}^{(1)}; \ldots; \boldsymbol{P}^{(m_1)}; \boldsymbol{O}]$, where $\boldsymbol{P}^{(k)} = [\boldsymbol{p}^{(k)}_1|\ldots|\boldsymbol{p}^{(k)}_{m_1}]$ and $\boldsymbol{p}^{(k)}_j \in \mathbb{Z}^{l_k}$. Let $\boldsymbol{H}' = \{h'_{i,j}\}_{i,j\in[m_1]}$.

For any $i, j \in [m_1]$, we have that

$$h'_{i,j} = \boldsymbol{g} \cdot \boldsymbol{p}^{(j)}_i$$

by the construction of $\boldsymbol{G}$. Hence, we set $\boldsymbol{p}^{(j)}_i$ to be a r-base decomposition of $h'_{i,j}$ and have that each coefficient in $\boldsymbol{p}^{(j)}_i$ is in $[0, r-1]$.

Length of S: The norm of $\boldsymbol{S}$ is $\max\{\|\boldsymbol{S}_1\|, \|\boldsymbol{S}_2\|\}$, where $\boldsymbol{S}_1 = [\boldsymbol{V}; \boldsymbol{P}]$ and $\boldsymbol{S}_2 = [\boldsymbol{D}; \boldsymbol{B}]$.

We start to estimate the norm of $\boldsymbol{S}_2$. Recall that $\boldsymbol{U} = \boldsymbol{G} + \boldsymbol{R}$ and $\boldsymbol{D} = \boldsymbol{U}\boldsymbol{B} = \boldsymbol{G}\boldsymbol{B} + \boldsymbol{R}\boldsymbol{B}$. We have that $\|\boldsymbol{G}\boldsymbol{B}\|^2 \leq \|\boldsymbol{W}\|^2 \leq 1$. We also have $\|\boldsymbol{R}\| \leq \sqrt{d}$ and thus $\|\boldsymbol{R}\boldsymbol{B}\| \leq (r+1)\sqrt{d}$. Hence, $\|\boldsymbol{D}\| \leq \|\boldsymbol{G}\boldsymbol{B}\| + \|\boldsymbol{R}\boldsymbol{B}\| \leq 1 + (r+1)\sqrt{d} \leq (r+2)\sqrt{d}$.

$$\|\boldsymbol{S}_2\|^2 \leq \|\boldsymbol{D}\|^2 + \|\boldsymbol{B}\|^2 \leq (r+2)^2 d + (r^2+1) \leq (r+2)^2(d+1).$$

Next, we estimate the norm of $\boldsymbol{S}_1$. Simply, we have that $\|\boldsymbol{P}\| \leq \sqrt{s} \cdot (r-1)$. Recall that $\boldsymbol{V} = \boldsymbol{R}\boldsymbol{P} - \boldsymbol{I}_{m_1}$. Hence, we have that

$$\|\boldsymbol{V}\| \leq \sqrt{d} \cdot (r-1)s + 1.$$

This indicates

$$\|\boldsymbol{S}_1\|^2 \leq \|\boldsymbol{V}\|^2 + \|\boldsymbol{P}\|^2 \leq (\sqrt{d}(r-1)s+1)^2 + (r-1)^2 s \leq 2dr^2s^2$$

for sufficiently large d and s.

Combining the above arguments, we have the upper bound $\sqrt{2d} \cdot rs$.

To obtain better upper bound, we use Hoeffding's inequality: Since $\boldsymbol{R}'$ is chosen from $\{-1, 0, +1\}^{d \times m_2}$ uniformly at random, for any S, which is any entry of $\boldsymbol{R}\boldsymbol{P}$, we have that $|S| \geq t\sqrt{s}$ with probability at most $2\exp(-2t^2/r^2)$. Setting $t = \omega(r\sqrt{\log n})$ and taking a union bound over all entries of $\boldsymbol{P}\boldsymbol{R}$, we have that $\|\boldsymbol{P}\boldsymbol{R}\| \leq t\sqrt{sd}$ with overwhelming probability. This shows that

$$\|\boldsymbol{S}_1\|^2 \leq \|\boldsymbol{V}\|^2 + \|\boldsymbol{P}\|^2 \leq (\sqrt{sd} \cdot t + 1)^2 + r^2 s = O(sdt^2)$$

and thus we have the upper bound $\sqrt{sd} \cdot \omega(\sqrt{\log n})$ with overwhelming probability.

10.3.3 The Second Construction

Theorem 10.3.3 (Alwen and Peikert [AP09]). *Let $\delta > 0$ and $r \geq 2$ be any constant. Let $m_1 = m_1(n)$, $m_2 = m_2(n)$, $m = m_1 + m_2$, and $q = q(n)$. Let l denote $\lceil \log_r(q-1) \rceil$. There is a probabilistic polynomial-time algorithm* ExtLattice2 *that, on input 1^n and uniformly random matrix $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n \times m_1}$, outputs a pair $(\boldsymbol{A} = [\boldsymbol{A}_1 | \boldsymbol{A}_2], \boldsymbol{S}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}^{m \times m}$. If $m_1 \geq d = (1+\delta) n \log q$ and $m_2 \geq m_1 \cdot l$,*

- *$\boldsymbol{A}$ is $(m_2 \cdot q^{-\delta n/2})$-uniform over $\mathbb{Z}_q^{n \times m}$,*
- *$\boldsymbol{S}$ is a basis of $\Lambda^\perp(\boldsymbol{A})$, and*
- *$\|\boldsymbol{S}\| \leq 2r\sqrt{m_1+1}$.*

Description: The basic idea is we make $\boldsymbol{G}$ contain the columns of $\boldsymbol{H}' = [\boldsymbol{h}'_1, \ldots, \boldsymbol{h}'_{m_1}] = \boldsymbol{H} - \boldsymbol{I}_{m_1}$. This drastically reduces the norm of $\boldsymbol{P}$. We again start with a construction of $\boldsymbol{B}$;

$$\boldsymbol{B} = \mathrm{diag}(\boldsymbol{T}_l, \ldots, \boldsymbol{T}_l, \boldsymbol{I}_{m_2 - m_1 \cdot l}).$$

We note that

$$\boldsymbol{B} = \mathrm{diag}(\boldsymbol{T}_l^{-1}, \ldots, \boldsymbol{T}_l^{-1}, \boldsymbol{I}_{m_2 - m_1 \cdot l}).$$

We next split $\boldsymbol{W}$ and $\boldsymbol{G}$ into $m_1 + 1$ matrices where $\boldsymbol{W} = [\boldsymbol{W}^{(1)}|\ldots|\boldsymbol{W}^{(m_1)}|\boldsymbol{O}]$, $\boldsymbol{G} = [\boldsymbol{G}^{(1)}|\ldots|\boldsymbol{G}^{(m_1)}|\boldsymbol{O}]$, and $\boldsymbol{W}^{(k)}, \boldsymbol{G}^{(k)} \in \mathbb{Z}^{m_1 \times l}$ for any $k \in [m_1]$. Let $\boldsymbol{W}^{(k)} = [\boldsymbol{w}_1^{(k)}, \ldots, \boldsymbol{w}_l^{(k)}]$ and $\boldsymbol{G}^{(k)} = [\boldsymbol{g}_1^{(k)}, \ldots, \boldsymbol{g}_l^{(k)}]$. Notice that $\boldsymbol{G}^{(k)} = \boldsymbol{W}^{(k)} \cdot \boldsymbol{T}_l^{-1}$. Let $\boldsymbol{w}_j^{(k)}$ be a reverse-order r-base decomposition of $\boldsymbol{h}'_k$, that is, $\boldsymbol{h}'_k = \sum_{j \in [l]} r^{l-j} \boldsymbol{w}_j^{(k)}$. Then,

$$\boldsymbol{g}_l^{(k)} = \sum_{j \in [l]} r^{l-j} \boldsymbol{w}_j^{(k)} = \boldsymbol{h}'_k.$$

Using this construction, making $\boldsymbol{GP} = \boldsymbol{H}' = \boldsymbol{H} - \boldsymbol{I}_{m_1}$ is again straightforward; The j-th column of $\boldsymbol{P}$ picks up $\boldsymbol{h}'_j$ in $\boldsymbol{G}$. More precisely, let $\boldsymbol{P} = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{m_1}]$ and let $\boldsymbol{p}_j = \boldsymbol{i}_{li}$ for $i \in [m_1]$.

Length of $\boldsymbol{S}$: The norm of $\boldsymbol{S}$ is $\max\{\|\boldsymbol{S}_1\|, \|\boldsymbol{S}_2\|\}$, where $\boldsymbol{S}_1 = [\boldsymbol{V}; \boldsymbol{P}]$ and $\boldsymbol{S}_2 = [\boldsymbol{D}; \boldsymbol{B}]$.

We start to estimate the norm of $\boldsymbol{S}_2$. Recall that $\boldsymbol{U} = \boldsymbol{G} + \boldsymbol{R}$ and $\boldsymbol{D} = \boldsymbol{UB} = \boldsymbol{GB} + \boldsymbol{RB}$. We have that $\|\boldsymbol{GB}\| = \|\boldsymbol{W}\| \leq \sqrt{m_1}(r-1)$. We also have $\|\boldsymbol{R}\| \leq \sqrt{d}$ and thus $\|\boldsymbol{RB}\| \leq (r+1)\sqrt{d}$. Hence, $\|\boldsymbol{D}\| \leq \|\boldsymbol{GB}\| + \|\boldsymbol{RB}\| \leq \sqrt{m_1}(r-1) + (r+1)\sqrt{d} \leq 2r\sqrt{m_1}$.

$$\|\boldsymbol{S}_2\|^2 \leq \|\boldsymbol{D}\|^2 + \|\boldsymbol{B}\|^2 \leq 4r^2 m_1 + r^2 + 1 \leq (2r)^2(m_1 + 1)$$

for sufficiently large m_1.

Next, we estimate the norm of $\boldsymbol{S}_1$. Simply, we have that $\|\boldsymbol{P}\| = 1$. Then, we also have $\|\boldsymbol{RP}\| \leq \sqrt{d}$. Hence, by the triangle inequality, we have that

$$\|\boldsymbol{V}\| \leq \sqrt{d} + 1$$

This indicates

$$\|\boldsymbol{S}_1\|^2 \leq \|\boldsymbol{V}\|^2 + \|\boldsymbol{P}\|^2 \leq d + 2\sqrt{d} + 1 + 1 \leq (\sqrt{d} + 2)^2.$$

Combining the above arguments, we have the upper bound $2r\sqrt{m_1 + 1}$.

10.3.4 The Third Construction

Theorem 10.3.4 ([AP09]). *Let $\delta > 0$ and $r \geq 2$ be any constants. Let $m_1 = m_1(n)$, $m_2 = m_2(n)$, $m = m_1 + m_2$, and $q = q(n)$ with q odd prime. There is a probabilistic polynomial-time algorithm* ExtLattice3 *that, on input 1^n and uniformly random matrix $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n \times m_1}$, outputs a pair $(\boldsymbol{A} = [\boldsymbol{A}_1|\boldsymbol{A}_2], \boldsymbol{S}) \in \mathbb{Z}_q^{n \times m_2} \times \mathbb{Z}^{m \times m}$. If $m_1 \geq d = (1+\delta)n \log q$ and $m_2 \geq (4 + 2\delta)n \log q$, there is a constant $C > 0$ such that*

- *$\boldsymbol{A}$ is $(m_2 \cdot q^{-\delta n/2})$-uniform over $\mathbb{Z}_q^m$,*
- *$\boldsymbol{S}$ is a basis of $\Lambda_q^{\perp}(\boldsymbol{A})$,*
- *$\|\boldsymbol{S}\| \leq Cn \log q$ with overwhelming probability, and*
- *$\|\tilde{\boldsymbol{S}}\| \leq 1 + C\sqrt{d} = O(\sqrt{n \log q})$ with overwhelming probability.*

Description: Roughly speaking, the construction is similar to the construction 1. The basic idea is implanting another matrix $\boldsymbol{M}$ into $\boldsymbol{G}$ to shorten the norm of $\tilde{\boldsymbol{S}}$, where rows of $\boldsymbol{M}$ is orthogonal.

We recall the definition of $\boldsymbol{B}$ in the construction 1;

$$\boldsymbol{B} = \mathrm{diag}(\boldsymbol{T}_{l_1}, \dots, \boldsymbol{T}_{l_{m_1}}, \boldsymbol{I}_{m_2 - s}),$$

where $l_i = \lceil \log_r c_i \rceil$ and $s = \sum_{i \in [m_1]} l_i$.

We next split $\boldsymbol{W}$ and $\boldsymbol{G}$ into $m_1 + 2$ matrices;

$$\boldsymbol{W} = [\boldsymbol{W}^{(1)} | \dots | \boldsymbol{W}^{(m_1)} | \boldsymbol{M} | \boldsymbol{O}] \text{ and } \boldsymbol{G} = [\boldsymbol{G}^{(1)} | \dots | \boldsymbol{G}^{(m_1)} | \boldsymbol{M} | \boldsymbol{O}],$$

where $\boldsymbol{W}^{(k)}, \boldsymbol{G}^{(k)} \in \mathbb{Z}^{m_1 \times l}$ for any $k \in [m_1]$. We define $\boldsymbol{M}$ later. As in the construction 1, let $w_{i,j}^{(k)} = 1$ when $i = k$ and $j = 1$, and 0 otherwise. Then, we have that $g_{i,j}^{(k)} = r^j$ when $i = k$, and 0 otherwise.

We next define $\boldsymbol{M} \in \mathbb{Z}^{m_1 \times w}$. Let w be the largest power of 2 in the range $[d, m_2 - 2n \log_r q]$. By the hypothesis, we have $m_2 - 2n \log_r q \geq 2d$. Thus, there is a power of w in the range. Notice that $w \geq m_2/2 - n \log_r q \geq m_2/4$. The matrix $\boldsymbol{M}$ is zero in all but its first d rows. The first d rows of $\boldsymbol{M}$ are set to be the C' multiple of d distinct rows of a square Hadamard matrix of dimension w. Note that, by the Sylvester construction, we always have a w by w Hadamard matrix $H_2^{\otimes \log_w}$, where $H_2 = [1, 1; -1, 1]$.

The matrix $\boldsymbol{P}$ is defined as the same way to the one in the construction 1.

Length of $\boldsymbol{S}$: The estimation of $\|\boldsymbol{S}\|$ is obtained by the almost same way to the one of the construction 1.

We omit the estimation of the length of $\tilde{\boldsymbol{S}}$, since this needs a somewhat complicated analysis on the singular values of random matrices. For the details, see the original paper [AP09].

10.4 The Sampling Algorithm

Theorem 10.4.1 ([GPV08])**.** *There is a probabilistic polynomial-time algorithm* SampleD *that, given a basis $\boldsymbol{T}$ of an n-dimensional lattice Λ, a parameter $s \geq \|\tilde{\boldsymbol{T}}\| \cdot \omega(\sqrt{\log m})$, and a center $\boldsymbol{c} \in \mathbb{R}^n$, outputs a sample from distribution that is statistically close to $D_{\Lambda,s,\boldsymbol{c}}$.*

We note that the algorithm SampleD is indeed the same as Klein's one, as Lyubashevsky pointed out.

The core of the algorithm used the acceptance–rejection method [vN51, Dev86]. Hence, we first review it.

10.4.1 The Acceptance–Rejection Method

We recall the acceptance–rejection method, the one of the basic methodologies for sampling from non-uniform distributions. This technique is formalized by von Neumann [vN51].

Suppose that we want to sample values according to a distribution f over S. Assume that we can sample values according to another distribution g over S. If, for any $x \in S$, we have $f(x) < cg(x)$ for some $c > 1$, we can use the acceptance–rejection method in order to sample from f. The algorithm is as follows:

1. Sample $x \leftarrow g$ and $u \leftarrow (0, 1)$.
2. If $u < f(x)/cg(x)$, output x. Otherwise output $\perp$.

In order to simplify the notation, we define $h(x) = f(x)/(cg(x))$ in this subsection. Let D^h denote the distribution of the output of the above algorithm using $h(x)$. $D^h(x)$ denotes the probability density function of the distribution D^h.

For a random variable $u \leftarrow (0, 1)$ and $x \in S$,

$$\Pr\left[u \leq \frac{f(x)}{cg(x)}\right] = \Pr\left[u \leq \frac{f(X)}{cg(X)} \mid X = x\right] = \frac{f(x)}{cg(x)} = h(x).$$

Thus,

$$D^h(x) = \begin{cases} \frac{f(x)}{cg(x)} \cdot g(x) = \frac{f(x)}{c} & (x \in S) \\ 1 - 1/c & (x = \perp) \end{cases}.$$

Therefore, the distribution f coincides with the distribution of the output conditioned on that the output is not $\perp$.

The correctness of the algorithm when repeated r-times is summarized as follows:

Lemma 10.4.2. *Consider the following algorithm:*

1. *Initialize $i \leftarrow 0$.*
2. *Sample $x \leftarrow g$ and $u \leftarrow (0, 1)$.*
3. *If $u < f(x)/(cg(x))$, output x. If $i \geq r$ output $\perp$. Otherwise go to Step 2.*

Let D denote the output distribution of the above algorithm. Then,

$$\Delta(D, f) = \left(1 - \frac{1}{c}\right)^r.$$

Proof. Since f coincides with the conditional distribution given that the output is not $\perp$, we have that

$$D(x) = \begin{cases} (1 - (1 - 1/c)^r)f(x) & (x \in S) \\ (1 - 1/c)^r & (x = \perp) \end{cases}.$$

To ease of notation, let δ denote $(1 - 1/c)^r$. We obtain that

$$\begin{aligned}\Delta(D, f) &= \frac{1}{2}\int_{x\in S\cup\{\perp\}} |D(x) - f(x)|\,dx \\ &= \frac{1}{2}\left(D(\perp) + \int_{x\in S} |D(x) - f(x)|\,dx\right) \\ &= \frac{1}{2}\left(\delta + \int_{x\in S} (1-\delta)f(x)dx\right) \\ &= \frac{1}{2}\cdot 2\delta = \delta,\end{aligned}$$

which completes the proof. □

10.4.2 Sampling over a One-Dimensional Lattice

The starting point is a sampling algorithm over a one-dimensional lattice.

Algorithm 1 SampleZ

```
Require: $1^n$, $s > 0$, $c \in \mathbb{R}$
Ensure: $x \leftarrow D_{\mathbb{Z},s,c}$
1: $x \leftarrow \mathbb{Z} \cap [c - st, c + st]$
2: $u \leftarrow [0, 1]$
3: if $\rho_s(x - c) \leq u$ then
4:     return $x$
5: else
6:     goto Step 1
7: end if
```

The following lemma ensures that the sample from $D_{\mathbb{Z},s,c}$ falls in the range $[c - st, c + st]$ with overwhelming probability if t is sufficiently large.

Lemma 10.4.3 ([GPV08]). *For any $\epsilon > 0$, any $s \geq \eta_\epsilon(\mathbb{Z})$, and any $t > 0$,*

$$\Pr_{x\leftarrow D_{\mathbb{Z},s,c}}[|x - c| \geq ts] \leq 2\cdot\frac{1+\epsilon}{1-\epsilon}\cdot\exp(-\pi t^2).$$

In particular, for $\epsilon \in (0, 1/2)$ and $t \geq \omega(\sqrt{\log n})$, the probability that $|x - c| \geq ts$ is negligible in n.

The correctness of the algorithm SampleZ is summarized as follows, which is obtained as the corollary of the above lemmas:

Lemma 10.4.4 ([GPV08]). *For any $0 < \epsilon < \exp(-\pi)$, any $s \geq \eta_\epsilon(\mathbb{Z})$, and $c \in \mathbb{R}$, and $t(n) = \omega(\sqrt{\log n})$,* SampleZ *terminates within $t(n)\cdot\omega(\log n)$ iterations with overwhelming probability, and its output distribution is statistically close to $D_{\mathbb{Z},s,c}$.*

10.4.3 Sampling over Arbitrary Lattice

The algorithm SampleD take a sample from $D_{\Lambda,s,\boldsymbol{c}}$. Its procedure is recursive one and can be interpreted as randomized nearest plane algorithm. Indeed, if we change the line 4 in SampleD, the algorithm is the nearest plane algorithm [Bab86].

Algorithm 2 SampleD

```
Require: a basis $T$ of an $n$-dimensional lattice $\Lambda$, a parameter $s > 0$, and a center $c \in \mathbb{R}^n$
Ensure: $x \leftarrow D_{\Lambda,s,c}$
1: $v_n \leftarrow \mathbf{0}$ and $c_n \leftarrow c$.
2: for $i = n$ to 1 do
3:     $c'_i \leftarrow \langle c_i, \tilde{t}_i\rangle / \|\tilde{t}_i\|^2 \in \mathbb{R}$ and $s'_i \leftarrow s/\|\tilde{t}_i\| > 0$
4:     $z_i \leftarrow D_{\mathbb{Z},s'_i,c'_i}$ (this is done by $z_i \leftarrow$ SampleZ$(1^n, s'_i, c'_i)$.)
5:     $c_{i-1} \leftarrow c_i - z_i t_i$ and $v_{i-1} \leftarrow v_i + z_i t_i$
6: end for
7: return $v_0$
```

From the construction $\boldsymbol{v}_{i-1} \leftarrow \boldsymbol{v}_i + z_i\boldsymbol{t}_i$, the output vector $\boldsymbol{v} = \boldsymbol{v}_0$ is a lattice vector.

For the consistency, we include the proof by Gentry et al.. They prepared two lemmas.

Lemma 10.4.5 (Lemma 4.4, [GPV08]). *For any* $(\boldsymbol{T}, s, \boldsymbol{c})$ *and any output* $\boldsymbol{v} = \boldsymbol{v}_0 = \sum_{i\in[n]} z_i\boldsymbol{t}_i \in \Lambda$ *of* SampleD,

$$\boldsymbol{v} - \boldsymbol{c} = \sum_{i\in[n]} (z_i - c'_i)\tilde{\boldsymbol{t}}_i.$$

Proof. For $i \in [n]$, let us define projections $\pi_j : \mathbb{R}^n \to \mathrm{span}(\boldsymbol{t}_1, \ldots, \boldsymbol{t}_i)$. We will show that for all $j = 0, \ldots, n$,

$$(\boldsymbol{v}_0 - \boldsymbol{v}_j) - \pi_j(\boldsymbol{c}_j) = \sum_{i\in[j]} (z_i - c'_i)\tilde{\boldsymbol{t}}_i.$$

It holds in the case where $j = 0$ trivially. Hence, suppose that it holds for $j = k - 1$ for some $k \in [n]$. By the construction, we have $\boldsymbol{v}_k = \boldsymbol{v}_{k-1} - z_k\boldsymbol{t}_k$ and $\boldsymbol{c}_k = \boldsymbol{c}_{k-1} + z_k\boldsymbol{t}_k$. In addition, we have that $\|\tilde{\boldsymbol{t}}_i\|^2 c'_i = \langle \boldsymbol{c}_k, \tilde{\boldsymbol{t}}_k\rangle$. Therefore, we have that

$$\begin{aligned}
\boldsymbol{v}_0 - \boldsymbol{v}_k - \pi_k(\boldsymbol{c}_k) &= \boldsymbol{v}_0 - (\boldsymbol{v}_{k-1} - z_k\boldsymbol{t}_k) - (\pi_{k-1}(\boldsymbol{c}_k) + c'_k\tilde{\boldsymbol{t}}_k) \\
&= (\boldsymbol{v}_0 - \boldsymbol{v}_{k-1}) + z_k\boldsymbol{t}_k - (\pi_{k-1}(\boldsymbol{c}_{k-1}) + \pi_{k-1}(z_k\boldsymbol{t}_k) + c'_k\tilde{\boldsymbol{t}}_k) \\
&= (\boldsymbol{v}_0 - \boldsymbol{v}_{k-1} - \pi_{k-1}(\boldsymbol{c}_{k-1})) + z_k(\boldsymbol{t}_k - \pi_{k-1}(\boldsymbol{t}_k)) - c'_k\tilde{\boldsymbol{t}}_k \\
&= (\boldsymbol{v}_0 - \boldsymbol{v}_{k-1} - \pi_{k-1}(\boldsymbol{c}_{k-1})) + (z_k - c'_k)\tilde{\boldsymbol{t}}_k \\
&= \sum_{i\in[k]} (z_i - c'_i)\tilde{\boldsymbol{t}}_i.
\end{aligned}$$

By the induction, we have this equation in $j = k$ and complete the proof. □

Lemma 10.4.6 (Lemma 4.5, [GPV08]). *For any input* $(\boldsymbol{T}, s, \boldsymbol{c})$ *and any output* $\boldsymbol{v} = \boldsymbol{v}_0 = \sum_{i\in[n]} \hat{z}_i \boldsymbol{t}_i \in \Lambda$ *of* SampleD*, the probability that* SampleD *outputs* $\boldsymbol{v}$ *is exactly*

$$\rho_{s,\boldsymbol{c}}(\boldsymbol{v}) \cdot \prod_{i\in[n]} \frac{1}{\rho_{s'_i,c'_i}(\mathbb{Z})}.$$

Proof. The vector $\boldsymbol{v}$ is output if every random choice $z_i = \hat{z}_i$ for $i = n, \ldots, 1$. Let E denote this event. For each i, the probability that $z_i = \hat{z}_i$, conditioned on $z_j = \hat{z}_j$ for all $j = n, \ldots, i+1$ is $D_{\mathbb{Z},s'_i,c'_i}(\hat{z}_i)$. Hence, the probability of E is

$$\prod_{i\in[n]} D_{\mathbb{Z},s'_i,c'_i}(\hat{z}_i) = \frac{\prod_{i\in[n]} \rho_{s'_i,c'_i}(\hat{z}_i)}{\prod_{i\in[n]} \rho_{s'_i,c'_i}(\mathbb{Z})}.$$

The numerator is

$$\prod_{i\in[n]} \rho_{s'_i,c'_i}(\hat{z}_i) = \prod_{i\in[n]} \rho_s((\hat{z}_i - c'_i)\cdot \|\tilde{\boldsymbol{t}}_i\|) = \rho_s\left(\sum_{i\in[n]} (\hat{z}_i - c'_i)\tilde{\boldsymbol{t}}_i\right) = \rho_s(\boldsymbol{v} - \boldsymbol{c}) = \rho_{s,\boldsymbol{c}}(\boldsymbol{v}),$$

where we use $s'_i = s/\|\tilde{\boldsymbol{t}}_i\|$ and the orthogonality of $\tilde{\boldsymbol{T}}$. This completes the proof. □

Finally, they proved the following theorem.

Theorem 10.4.7 (Theorem 4.1, [GPV08]). *Given a basis* $\boldsymbol{T}$ *of an n-dimensional lattice* Λ*, a parameter* $s \geq \|\tilde{\boldsymbol{T}}\| \cdot \omega(\sqrt{\log n})$*, and a center* $\boldsymbol{c} \in \mathbb{R}^n$*, the algorithm* SampleD *outputs a sample from a distribution that is statistically close to* $D_{\Lambda,s,\boldsymbol{c}}$*.*

Proof. Let $s \geq \|\tilde{\boldsymbol{T}}\| \cdot g(n)$ for some $g(n) = \omega(\sqrt{\log n})$. Then, we have that $s'_i = s/\|\tilde{\boldsymbol{t}}_i\| \geq g(n)$. By Lemma 2.1.7, we have that $\eta_\epsilon(\mathbb{Z}) \leq \tilde{bl}(\mathbb{Z})\cdot \sqrt{\log(2n(1+1/\epsilon))/\pi} \leq \sqrt{\log(2n(1+1/\epsilon))/\pi}$. Thus, by setting $\epsilon(n) = 2^{-O(g^2(n))} = \mathsf{negl}(n)$ appropriately, we have $g(n) \geq \eta_\epsilon(\mathbb{Z})$ and each $s'_i \geq \eta_\epsilon(\mathbb{Z})$. Hence, the SampleZ implements the oracle $D_{\mathbb{Z},s'_i,c'_i}$ within negligible statistical distance.

We show that SampleD using $D_{\mathbb{Z},s',c'}$ samples to withing negligible statistical distance of $D_{\Lambda,s,\boldsymbol{c}}$. Let $Q = \rho_{s,\boldsymbol{c}}(\Lambda)$. Then, the probability function of $\boldsymbol{v}$ under $D_{\Lambda,s,\boldsymbol{c}}$ is $\rho_{s,\boldsymbol{c}}(\boldsymbol{v})/Q$. Meanwhile, Lemma 2.1.10 implies that

$$\rho_{s'_i,c'_i}(\mathbb{Z}) \in [\tfrac{1-\epsilon}{1+\epsilon}, 1] \cdot \rho_{s'_i}(\mathbb{Z})$$

for any value c'_i. By the above lemma, for every $\boldsymbol{v} \in \Lambda$, the probability that SampleD outputs $\boldsymbol{v}$ is in the range

$$R^{-1} \cdot [1, (\tfrac{1+\epsilon}{1-\epsilon})^n] \cdot \rho_{s,\boldsymbol{c}} \subseteq R^{-1} \cdot [1, 1+\epsilon'] \cdot \rho_{s,\boldsymbol{c}}(\boldsymbol{v}),$$

where $R = \prod_{i\in[n]} \rho_{s'_i}(\mathbb{Z})$ and $\epsilon'(n)$ is some negligible function of n. This shows that $R \in [1, 1+\epsilon']Q$ and the distance is at most $\epsilon'/2$. □

10.5 Lattice-Based Collision-Resistant Preimage Sampleable Function

We now return to the lattice-based collision-resistant PSFs. Gentry et al. showed the scheme combining Ajtai's trapdoor generation, Ajtai's hash functions, and the sampling algorithm in the previous section (Section 10.4) is indeed collision-resistant PSFs (statistically).

Scheme 10.5.1 (LPSF [GPV08])**.**

TrapGen(1^n)**:** The same as ExtLattice3(1^n) in Section 10.3. It outputs $(\boldsymbol{A}, \boldsymbol{T})$, where $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ is statistically close to uniform and $\boldsymbol{T} \subset \Lambda_q^{\perp}(\boldsymbol{A})$ is a good basis with $\|\tilde{\boldsymbol{T}}\| \leq L = O(\sqrt{n \lg q})$. The matrix $\boldsymbol{A}$ defines the function $h_{\boldsymbol{A}}(\cdot)$. This function is defined as $h_{\boldsymbol{A}}(\boldsymbol{e}) = \boldsymbol{A}\boldsymbol{e} \bmod q$ with domain $D_n = \{\boldsymbol{e} \in \mathbb{Z}^m : \|\boldsymbol{e}\| \leq s\sqrt{m}\}$ and range $R_n = \mathbb{Z}_q^n$.

SampleDom(1^n)**:** The input distribution is $D_{\mathbb{Z}^m,s}$. Hence, this algorithm invokes SampleD with inputs $\boldsymbol{I}_m$, s, and $\boldsymbol{0}$ and outputs the sample.

SamplePre($\boldsymbol{A}, \boldsymbol{T}, s, \boldsymbol{u}$)**:** The algorithm samples from $h_{\boldsymbol{A}}^{-1}(\boldsymbol{u})$ as follows: It generates $\boldsymbol{t} \in \mathbb{Z}^m$ such that $\boldsymbol{A}\boldsymbol{t} = \boldsymbol{u} \bmod q$ by standard algebra, samples $\boldsymbol{v} \leftarrow D_{\Lambda_q^{\perp}(\boldsymbol{A}),s,-\boldsymbol{t}}$ by SampleD($\boldsymbol{T}, s, -\boldsymbol{t}$), and outputs $\boldsymbol{e} = \boldsymbol{t} + \boldsymbol{v}$.

We start with several lemmas.

Lemma 10.5.2 (Regev, [Reg09])**.** *Let $m \geq 2n \log q$. Then for all but an at most q^{-n} fraction of $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, the subset-sums of the columns of $\boldsymbol{A}$ generate $\mathbb{Z}_q^n$. That is, for every $\boldsymbol{u} \in \mathbb{Z}_q^n$, there is an error vector $\boldsymbol{e} \in \{0,1\}^m$ such that $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q$.*

Lemma 10.5.3 (Lemma 5.2, [GPV08])**.** *Assume the columns of $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ generate $\mathbb{Z}_q$ and let $\epsilon \in (0, 1/2)$ and $s \geq \eta_\epsilon(\Lambda_q^{\perp}(\boldsymbol{A}))$. Then for $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^m,s}$, the distribution of the syndrome $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \bmod q$ is within statistical distance 2ϵ of uniform over $\mathbb{Z}_q^n$.*

Furthermore, fix $\boldsymbol{u}$ and $\boldsymbol{t} \in \mathbb{Z}^m$ be an arbitrary solution to $\boldsymbol{A}\boldsymbol{t} = \boldsymbol{u} \bmod q$. Then the conditional distribution of $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^m,s}$ given $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q$ is exactly $\boldsymbol{t} + D_{\Lambda_q^{\perp}(\boldsymbol{A}),s,-\boldsymbol{t}}$.

Lemma 10.5.4 ([GPV08])**.** *Let n and q be positive integers with q prime, and let $m \geq 2n \log q$. Then for all but an at most q^{-n} fraction of $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, we have $\lambda_1^{\infty}(\Lambda_q(\boldsymbol{A})) \geq q/4$.*

In particular, for such $\boldsymbol{A}$ and for any $\omega(\sqrt{\log m})$ function, there is a negligible function $\epsilon(m)$ such that $\eta_\epsilon(\Lambda_q^{\perp}(\boldsymbol{A})) \leq \omega(\sqrt{\log m})$.

Corollary 10.5.5 ([GPV08])**.** *Let n and q be positive integers with q prime, and let $m \geq 2n \log q$. Then for all but an at most $2q^{-n}$ fraction of $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and for any $s \geq \omega(\sqrt{\log m})$, the distribution of the syndrome $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \bmod q$ is statistically close to uniform over $\mathbb{Z}_q^n$, where $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^m,s}$.*

By using these lemmas, we can proof the security of LPSF.

Theorem 10.5.6 ([GPV08]). *The above scheme* LPSF *is collision resistant if* $\mathrm{SIS}_{q,m,2s\sqrt{m}}$ *is hard and* $m \geq (5+3\delta)\log q$ *for some constant* $\delta > 0$.

Proof. We note that $s \geq L \cdot \omega(\sqrt{\log m}) \geq \eta_\epsilon(\Lambda_q^\perp(\boldsymbol{A}))$ for some negligible $\epsilon(n)$ by Lemma 2.1.7 since $L \geq \|\tilde{\boldsymbol{T}}\| \geq \tilde{bl}(\Lambda_q^\perp(\boldsymbol{A}))$.

Next, a sample $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^m,s}$ falls into D_n except with negligible probability by Lemma 2.1.9. Furthermore, for all but a q^{-n} fraction of $\boldsymbol{A}$, $h_{\boldsymbol{A}}(\boldsymbol{e})$ is statistically close to uniform over $R_n = \mathbb{Z}_q^n$ by Corollary 10.5.5.

The preimage sampleable property follows from $s \geq \|\tilde{\boldsymbol{T}}\| \cdot \omega(\sqrt{\log m})$, Lemma 10.5.3, and the correctness of $\mathsf{SampleD}$ (Theorem 10.4.7); The samples from a distribution is statistically close to $D_{\Lambda_q^\perp(\boldsymbol{A}),s,-\boldsymbol{t}}$ and the conditional distribution of $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^m,s}$ given $\boldsymbol{A}\boldsymbol{e} \equiv \boldsymbol{u} \pmod{q}$ is exactly $\boldsymbol{t} + D_{\Lambda_q^\perp(\boldsymbol{A}),s,-\boldsymbol{t}}$.

The collision resistance property immediately follows from the hardness of $\mathrm{SIS}_{q,m,2s\sqrt{m}}$.

The preimage min-entropy is at least $m-1$. This follows the fact that the preimages are distributed according to $\boldsymbol{t} + D_{\Lambda_q^\perp(\boldsymbol{A}),s,-\boldsymbol{t}}$ and the min-entropy of $D_{\Lambda_q^\perp(\boldsymbol{A}),s,-\boldsymbol{t}}$ is at least $m-1$ (see Lemma 2.1.14). □

10.6 Ideal-Lattice Version of the Alwen-Peikert Construction

In order to obtain the ideal-lattice-based collision-resistant PSFs, we need to an ideal-lattice version of the Ajtai algorithm, which is proposed by Sthelé et al. [SSTX09]. The core idea is dividing each matrices into n by n submatrices and letting them to be rotation matrices corresponding to polynomials in $R_{\mathbf{f},q}$.

Quick remainders on polynomials and rings: For a monic polynomial $\mathbf{f}$ of degree n which is irreducible over $\mathbb{Z}$, we define $R_{\mathbf{f}} = \mathbb{Z}[x]/\langle \mathbf{f} \rangle$. For an integer q and such $\mathbf{f}$, $R_{\mathbf{f},q}$ denotes $\mathbb{Z}_q[x]/\langle \mathbf{f} \rangle$.

The number of units in $R_{\mathbf{f},q}$ plays an important role for regularity (see Section 4.4.2). Hence, we quickly analyze the number. For any integer q and any monic polynomial $\mathbf{f}$, we have $|R_{\mathbf{f},q}^*|/|R_{\mathbf{f},q}| \geq \prod_{i\in[t]}(1-(\phi(q)/q)^{\deg(\mathbf{f}_i)})$, where $\phi(\cdot)$ is Euler's phi function and $\mathbf{f} = \prod_{i\in[t]} \mathbf{f}_i$ is the factorization of $\mathbf{f}$ over $\mathbb{Z}_q$. If $\mathbf{f}$ is invertible over $\mathbb{Z}_q$, we have $\geq 1-(\phi(q)/q)^n$. If q is an odd prime and $\mathbf{f}$ is completely split over $\mathbb{Z}_q$, we have $\geq (1-1/q)^n$. If q is an odd prime, we have $|R_{\mathbf{f},q}^*|/|R_{\mathbf{f},q}| \geq \prod_{i\in[t]}(1-q^{-\deg(\mathbf{f}_i)})$.

10.6.1 The Stehlé–Steinfeld–Tanaka–Xagawa Construction

Stehlé, Steinfeld, Tanaka, and Xagawa [SSTX09] proposed the ideal-lattice version of the Alwen–Peikert construction.

Let $M^\perp(\breve{\boldsymbol{a}})$ denote the module $\{\breve{\boldsymbol{e}} \in R_{\mathbf{f}}^m \mid \breve{\boldsymbol{a}}\breve{\boldsymbol{e}} \equiv 0 \pmod{q}\}$. We will construct the basis $\boldsymbol{T}$ of the module. Let us consider the following construction: We first compute a basis $\boldsymbol{F}$ of $M^{\perp}(\check{\boldsymbol{a}})$. This basis $\boldsymbol{F}$ is not short. Hence, we then construct a unimodular matrix $\boldsymbol{Q}$ such that a basis $\boldsymbol{S} = \boldsymbol{F}\boldsymbol{Q}$ is short. Precisely, $\boldsymbol{S}$ has the following form as in the Alwen–Peikert construction:

$$\begin{bmatrix} \bar{\boldsymbol{a}}_1 & \bar{\boldsymbol{a}}_2 \end{bmatrix}\begin{bmatrix} \boldsymbol{V} & \boldsymbol{D} \\ \boldsymbol{P} & \boldsymbol{B} \end{bmatrix} = \boldsymbol{O} \text{ and } \underbrace{\begin{bmatrix} \boldsymbol{V} & \boldsymbol{D} \\ \boldsymbol{P} & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{S}} = \underbrace{\begin{bmatrix} \boldsymbol{H} & \boldsymbol{U} \\ \boldsymbol{O} & \boldsymbol{I}_{m_2} \end{bmatrix}}_{\boldsymbol{F}}\underbrace{\begin{bmatrix} -\boldsymbol{I}_{m_1} & \boldsymbol{O} \\ \boldsymbol{P} & \boldsymbol{B} \end{bmatrix}}_{\boldsymbol{Q}}$$

By construction, we have that

$$\bar{\boldsymbol{a}}_1\boldsymbol{H} + \bar{\boldsymbol{a}}_2\boldsymbol{O} \equiv \bar{\boldsymbol{0}} \pmod q \text{ and } \bar{\boldsymbol{a}}_1\boldsymbol{U} + \bar{\boldsymbol{a}}_2\boldsymbol{I}_{m_2} \equiv \bar{\boldsymbol{0}} \pmod q.$$

Thus,

$$\bar{\boldsymbol{a}}_1\boldsymbol{H} \equiv \bar{\boldsymbol{0}} \pmod q \text{ and } \bar{\boldsymbol{a}}_2 \equiv -\bar{\boldsymbol{a}}_1\boldsymbol{U} \pmod q.$$

In the Alwen–Peikert construction they set $\boldsymbol{H}$ to be the Hermite normal form of $\Lambda_q^{\perp}(\boldsymbol{A}_1)$, however we cannot define the Hermite normal form of a basis $M^{\perp}(\check{\boldsymbol{a}}_1)$ in the case where $\mathbf{f}$ is reducible over $\mathbb{Z}_q$. This is overcome later and we suppose some matrix $\boldsymbol{H}$, a basis of $M^{\perp}(\check{\boldsymbol{a}}_1)$. By setting $\boldsymbol{U} = \boldsymbol{G} + \boldsymbol{R}$, with $\boldsymbol{G}$ to be defined later on and $\boldsymbol{R}$ a random matrix, we have that $\bar{\boldsymbol{a}}_2$ is almost uniformly random by Micciancio's regularity lemma instead of direct applying the leftover hash lemma. More precisely, the columns of $\boldsymbol{R}$ is chosen from $(\{-1,0,1\}^n)^d \times (\{0\}^n)^{m_1-d}$.

According to the structure of $\boldsymbol{S}$, we have that

$$\boldsymbol{D} = (\boldsymbol{G} + \boldsymbol{R})\boldsymbol{B} \text{ and } \boldsymbol{V} = -\boldsymbol{H} + (\boldsymbol{G} + \boldsymbol{R})\boldsymbol{P}.$$

The matrix $\boldsymbol{G}$ will be designed to $\boldsymbol{G}\boldsymbol{P} = \boldsymbol{H}_2 - \boldsymbol{I}_d$. So, we let $\boldsymbol{V} = \boldsymbol{R}\boldsymbol{P} - \boldsymbol{I}_{m_1}$. Note that $\boldsymbol{D} = \boldsymbol{G}\boldsymbol{B} + \boldsymbol{R}\boldsymbol{B}$. We let $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{B}$.

Formally, we will show the following theorem.

Theorem 10.6.1 (Main Lemma, rearranged, [SSTX09]). *There are probabilistic polynomial-time algorithms with the following properties. They takes an odd prime q and integers n, σ, d, m_1, and m_2. They also takes a monic and irreducible polynomial $\mathbf{f} \in \mathbb{Z}[x]$ of degree n and random polynomials $\bar{\boldsymbol{a}}_1 \in R_{\mathbf{f},q}^{m_1}$, where $R_{\mathbf{f},q} = \mathbb{Z}_q[x]/\langle \mathbf{f} \rangle$. Let $\mathbf{f} = \prod_{i\in[t]} \mathbf{f}_i$ be the factorization of f over $\mathbb{Z}_q$. We let $\kappa = \lceil 1 + \log q \rceil$, $\Delta = \sqrt{-1 + \prod_{i\in[t]}(1 + (\frac{q}{3^d})^{\deg(\mathbf{f}_i)})}$, and $m = m_1 + m_2$. The algorithms succeed with probability $p_{\mathrm{succ}} \geq 1 - p_{\mathrm{fail}}$ over $\bar{\boldsymbol{a}}_1$, where $p_{\mathrm{fail}} = \left(1 - \prod_{i\in[t]}(1 - q^{-\deg(\mathbf{f}_i)})\right)^{\sigma}$. When they do,*

1. *The distance to uniformity of $\bar{\mathbf{a}}$ is at most $p_{\mathrm{fail}} + m_2\Delta$.*
2. *The quality of $\boldsymbol{S}$ is as follows:*
   - *If $m_1 \geq \max\{\sigma, \kappa, d\}$ and $m_2 \geq \kappa$, then $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})\| \leq \mathrm{EF}(\mathbf{f}, 2) \cdot \sqrt{2}\kappa d^{1/2} n^{3/2}$. Additionally, $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})\| \leq \mathrm{EF}(\mathbf{f}, 2) \cdot \sqrt{3a\kappa d} \cdot n$ with probability $1 - 2^{-a+O(\log nm_1 d)}$ for a super-logarithmic function $a = a(n) = \omega(\log n)$.*
   - *If $m_1 \geq \max\{\sigma, \kappa, d\}$ and $m_2 \geq \kappa \cdot m_1$, then $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})\| \leq \mathrm{EF}(f, 2) \cdot (4\sqrt{nd} + 3)$.*

3. *In particular, for* $\mathbf{f} = x^{2^k} + 1$ *with* $k \geq 2$ *and a prime* q *with* $q \equiv 3 \pmod 8$*, the following holds:*
   - *We can set* $\sigma = 1$ *and* $r = \lceil 1 + \log_3 q \rceil$*. Then, the error probability is* $p_{\text{fail}} = q^{-\Omega(n)}$ *and the parameter* Δ *is* $2^{-\Omega(n)}$*.*
   - *If* $m_1, m_2 \geq \kappa$*, then* $\|\text{Rot}_{\mathbf{f}}(\boldsymbol{S})\| \leq \sqrt{6a\kappa d} \cdot n = O(\sqrt{a}n \log q)$ *with probability* $1 - 2^{-a+O(\log(nm_1 \log q))}$ *for a super-logarithmic function* $a = a(n) = \omega(\log n)$*.*
   - *If* $m_1 \geq \kappa$ *and* $m_2 \geq \kappa \cdot m_1$*, then* $\|\text{Rot}_{\mathbf{f}}(\boldsymbol{S})\| \leq \sqrt{2}(4\sqrt{nd}+3) = O(\sqrt{n \log q})$*.*

For the sake of notation, we name the algorithms ExtIdLattice1 for the case where $m_2 \geq \kappa$ and ExtIdLattice2 for the case where $m_2 \geq m_1\kappa$.

We follow the Alwen–Peikert construction in which r is fixed to 2. Let $m_1 \geq \kappa = \lceil 1 + \log q \rceil$ and $m = m_1 + m_2$. Given random polynomials $\check{\boldsymbol{a}}_1 = (\mathbf{a}_1, \ldots, \mathbf{a}_{m_1})$, we should construct random polynomials $\check{\boldsymbol{a}}_2$ with a basis $\boldsymbol{S}$ of $M^{\perp}(\check{\boldsymbol{a}})$, where $\check{\boldsymbol{a}} = [\check{\boldsymbol{a}}_1 | \check{\boldsymbol{a}}_2]$. We need an Hermite normal form of $M^{\perp}(\check{\boldsymbol{a}}_1)$. However, if $\mathbf{f}$ is not invertible in $\mathbb{Z}_q$, we cannot define the Hermite normal form over $R_{\mathbf{f},q}$. This circumvent is overcome with a simple idea: Use of an HNF-like matrix.

Construction of $\boldsymbol{H}$ without Hermite Normal Forms: At first, we note that the one of $\mathbf{a}_i$ is in $R_{\mathbf{f},q}^*$ with probability at least $1 - p_{\text{fail}}$, since $m_1 \geq \sigma$. Let i^* denote such index. For now, we set $i^* = 1$ for simplicity. Although we have no definition for the HNF, we can construct the following HNF-like basis $\boldsymbol{H} = \{\mathbf{h}_{i,j}\}_{i,j \in [m_1]}$ of $M^{\perp}(\check{\boldsymbol{a}}_1)$: The first column is $q\check{\boldsymbol{\imath}}_1$ and the i-th column is $\mathbf{h}_i \check{\imath}_1 + \check{\imath}_i$ for $i = 2, \ldots, m_1$, where $\check{\imath}_i$ is a column vector in $R_{\mathbf{f}}^{m_1}$ such that the i-th element is 1 and others are 0, and $\mathbf{h}_i = -\mathbf{a}_i \otimes \mathbf{a}_1^{-1} \bmod q$ such that $\mathbf{h}_i \in [0, q)^n$. Illustratively, we have

$$\boldsymbol{H} = \begin{bmatrix} q & \mathbf{h}_2 & \ldots & \mathbf{h}_{m_1-1} & \mathbf{h}_{m_1} \\ & 1 & & & \\ & & \ddots & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}.$$

By the construction, we have the following lemma.

Lemma 10.6.2. *The matrix* $\boldsymbol{H}$ *is a basis of* $M^{\perp}(\check{\boldsymbol{a}}_1) \subseteq R_{\mathbf{f}}^{m_1}$*.*

Proof. Let $\check{\boldsymbol{h}}_i$ denote the i-th column of $\boldsymbol{H}$. By the definition of $\boldsymbol{H}$, $\check{\boldsymbol{a}}_1 \boldsymbol{H} \equiv \check{\mathbf{0}} \pmod q$. Hence, $\boldsymbol{H} \subseteq M^{\perp}(\check{\boldsymbol{a}}_1)$. It is obvious that $\check{\boldsymbol{h}}_1, \ldots, \check{\boldsymbol{h}}_{m_1}$ are linearly independent over $R_{\mathbf{f}}$.

In order to verify that $\boldsymbol{H}$ is a basis of the module, we need to show that, for each $\check{\boldsymbol{y}} \in M^{\perp}(\check{\boldsymbol{a}}_1)$, there exists a vector $\check{\boldsymbol{c}} \in R_{\mathbf{f}}^{m_1}$ such that $\check{\boldsymbol{y}} = \boldsymbol{H}\check{\boldsymbol{c}}$. Since the columns of $\boldsymbol{H}$ are linearly independent, there exists $\check{\boldsymbol{c}} \in (\mathbb{Q}[x]/\langle \mathbf{f} \rangle)^{m_1}$ such that $\check{\boldsymbol{y}} = \boldsymbol{H}\check{\boldsymbol{c}}$. Hence, it is remaining to show $\check{\boldsymbol{c}} \in R_{\mathbf{f}}^{m_1}$. The equation $\check{\boldsymbol{y}} = \boldsymbol{H}\check{\boldsymbol{c}}$ implies that $\mathbf{y}_1 = q\mathbf{c}_1 + \sum_{i=2}^{m_1} \mathbf{h}_i \otimes \mathbf{c}_i$ and $\mathbf{y}_i = \mathbf{c}_i$ for $i = 2, \ldots, m_1$. Since $\mathbf{y}_i \in R_{\mathbf{f}}$ for $i \in [m_1]$, we have that $\mathbf{c}_i \in R_{\mathbf{f}}$ for $i = 2, \ldots, m_1$ and $q\mathbf{c}_1 = \mathbf{y}_1 - \sum_{i=2}^{m_1} \mathbf{h}_i \otimes \mathbf{y}_i \in R_{\mathbf{f}}$. By the assumption on $\check{\boldsymbol{y}} = (\mathbf{y}_1, \ldots, \mathbf{y}_{m_1})$, we have that $\sum_{i\in[m_1]} \mathbf{a}_i \otimes \mathbf{y}_i \equiv 0 \pmod{q}$. In addition recall that the definition of $\mathbf{h}_i \equiv -\mathbf{a}_i \otimes \mathbf{a}_1^{-1} \pmod{q}$. Hence, we have that

$$\begin{aligned} q\mathbf{c}_1 &\equiv \mathbf{y}_1 - \sum_{i=2}^{m_1} \mathbf{h}_i \otimes \mathbf{y}_i \equiv \mathbf{y}_1 + \mathbf{a}_1^{-1} \otimes \left(\sum_{i=2}^{m_1} \mathbf{a}_i \otimes \mathbf{y}_i \right) \\ &\equiv \mathbf{a}_1^{-1} \otimes \left(\mathbf{a}_1 \otimes \mathbf{y}_1 + \sum_{i=2}^{m_1} \mathbf{a}_i \otimes \mathbf{y}_i \right) \equiv \mathbf{a}_1^{-1} \otimes 0 \equiv 0 \pmod{q}. \end{aligned}$$

Thus, $\mathbf{c}_1 \in R_{\mathbf{f}}$ and we conclude that $\boldsymbol{H}$ is a basis of $M^{\perp}(\check{\boldsymbol{a}}_1)$. □

Next, we consider the case where $i^* \neq 1$. In this case, we swap the columns 1 and i^* of $\boldsymbol{A}_1$ and call it $\check{\boldsymbol{a}}_1'$. Applying the method above, we obtain $\boldsymbol{H}'$, a basis of $M^{\perp}(\check{\boldsymbol{a}}_1')$. Again, swap the columns and the rows of $\boldsymbol{H}'$ we obtain $\boldsymbol{H}$, a basis of $M^{\perp}(\check{\boldsymbol{a}}_1)$. In the following, we denote by i^* the index i such that $\mathbf{a}_i \in R_{\mathbf{f},q}^*$ and $\mathbf{h}_{i,i} = q$.

Preliminaries of the constructions: Hereafter, we set $\boldsymbol{W} = \boldsymbol{G}\boldsymbol{B}$. We often use the matrix $\boldsymbol{T}_\kappa = \{\mathbf{t}_{i,j}\} \in R_{\mathbf{f}}^{\kappa\times\kappa}$, where $\mathbf{t}_{i,i} = 1$, $\mathbf{t}_{i,i+1} = -2$, and all other $t_{i,j}$'s are 0. Illustratively,

$$\boldsymbol{T}_\kappa = \begin{array}{c|ccccc} & 0 & 1 & 2 & \cdots & \kappa-1 \\ \hline 0 & 1 & -2 & & & \\ 1 & & 1 & -2 & & \\ 2 & & & 1 & \ddots & \\ \vdots & & & & \ddots & -2 \\ \kappa-1 & & & & & 1 \end{array}, \qquad \boldsymbol{T}_\kappa^{-1} = \begin{array}{c|ccccc} & 0 & 1 & 2 & \cdots & \kappa-1 \\ \hline 0 & 1 & 2 & 2^2 & \ldots & 2^{\kappa-1} \\ 1 & & 1 & 2 & \ldots & 2^{\kappa-2} \\ 2 & & & 1 & \ddots & \vdots \\ \vdots & & & & \ddots & 2 \\ \kappa-1 & & & & & 1 \end{array}.$$

We can verify $\boldsymbol{T}_\kappa^{-1}$ is the inverse of $\boldsymbol{T}_\kappa$ by a multiplication of them as in the Alwen–Peikert construction.

10.6.2 An Analog of the Alwen–Peikert Construction 1

We start with a construction of $\boldsymbol{B}$; we set

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{T}_\kappa & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{m_2-\kappa} \end{bmatrix}, \qquad \boldsymbol{B}^{-1} = \begin{bmatrix} \boldsymbol{T}_\kappa^{-1} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I}_{m_2-\kappa} \end{bmatrix},$$

We next set $\boldsymbol{W} = [\check{\boldsymbol{\imath}}_{i^*}\ \check{\mathbf{0}} \ldots \check{\mathbf{0}}] \in R_{\mathbf{f}}^{m_1\times m_2}$. By the construction of $\boldsymbol{W}$, we have that $\boldsymbol{G} = [\check{\boldsymbol{\imath}}_{i^*}\ 2\check{\boldsymbol{\imath}}_{i^*} \ldots 2^{\kappa-1}\check{\boldsymbol{\imath}}_{i^*}\ \check{\mathbf{0}} \ldots \check{\mathbf{0}}] \in R_{\mathbf{f}}^{m_1\times m_2}$, since $\boldsymbol{G} = \boldsymbol{W}\boldsymbol{B}^{-1}$. Notice that the columns of $\boldsymbol{H} - \boldsymbol{I}_{m_1}$ except the i^*-th row are all zero vectors, while the i^*-th row is $[\mathbf{h}_1, \ldots, \mathbf{h}_{i^*-1}, \mathbf{h}_{i^*} - 1, \mathbf{h}_{i^*+1}, \ldots, \mathbf{h}_{m_1}]$, where $\mathbf{h}_{i^*} - 1 = q - 1$. Using this construction and the above fact, making $\boldsymbol{G}\boldsymbol{P} = \boldsymbol{H} - \boldsymbol{I}_{m_1}$ is straightforward; Let $\boldsymbol{P} = \{\mathbf{p}_{i,j}\} \in R_{\mathbf{f}}^{m_2\times m_1}$. We let $\mathbf{p}_{i,j} \in \{0,1\}^n$ for $i \in [\kappa]$ and $j \in [m_1]$ such that $\mathbf{h}_j = \sum_{i\in[\kappa]} 2^{i-1}\mathbf{p}_{i,j}$. In addition, for $i = \kappa + 1, \ldots, m_2$ and for $j \in [m_1]$, let $\mathbf{p}_{i,j} = 0$. We then have $\boldsymbol{G}\boldsymbol{P} = \boldsymbol{H} - \boldsymbol{I}_{m_1}$.

Length of $\boldsymbol{S}$: The norm of $\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})$ is $\max\{\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S}_1)\|, \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S}_2)\|\}$, where $\boldsymbol{S}_1 = [\boldsymbol{V}; \boldsymbol{P}]$ and $\boldsymbol{S}_2 = [\boldsymbol{D}; \boldsymbol{B}]$. The estimations are the same as that in the Alwen–Peikert construction, we omit them.

10.6.3 An Analog of the Alwen–Peikert Construction 2

The idea in [AP09] is to have $\boldsymbol{G}$ contain the columns of $\boldsymbol{H}-\boldsymbol{I}_{m_1}$. This helps decrease the norms of the columns of $\boldsymbol{P}$ and $\boldsymbol{V}$.

To do so, we again start with construction of $\boldsymbol{B}$. Recall the inequality $m_2 \geq \kappa m_1$. Define $\boldsymbol{B}$ be the matrix of the form

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{T}_\kappa & & & \\ & \ddots & & \\ & & \boldsymbol{T}_\kappa & \\ & & & \boldsymbol{I}_{m_2-\kappa m_1} \end{bmatrix}, \quad \boldsymbol{B}^{-1} = \begin{bmatrix} \boldsymbol{T}_\kappa^{-1} & & & \\ & \ddots & & \\ & & \boldsymbol{T}_\kappa^{-1} & \\ & & & \boldsymbol{I}_{m_2-\kappa m_1} \end{bmatrix}.$$

Let $\check{\boldsymbol{h}}'_k$ denote the k-th column of $\boldsymbol{H} - \boldsymbol{I}_{m_1}$. Recall that $\check{\boldsymbol{h}}'_k = \mathbf{h}_k \otimes \check{\boldsymbol{\imath}}_{i^*}$ for some $\mathbf{h}_k$ in $[0, q-1)^n$.

Let us consider $\boldsymbol{G}_k = \{\mathbf{g}_{i,j}^{(k)}\}$ and $\boldsymbol{W}_k = \{\mathbf{w}_{i,j}^{(k)}\}$ in $R_{\mathbf{f}}^{m_1 \times \kappa}$ for $k \in [m_1]$. We have $\boldsymbol{G}_k = \boldsymbol{W}_k \cdot \boldsymbol{T}_\kappa^{-1}$ contain $\check{\boldsymbol{h}}'_k$. In order to do so, we let $\mathbf{w}_{i^*,j}^{(k)} \in \{0,1\}^n$ for $j \in [\kappa]$ such that $\mathbf{h}_k = \sum_{j \in [\kappa]} 2^{\kappa-j} \mathbf{w}_{i^*,j}^{(k)}$ and $\mathbf{w}_{i,j}^{(k)} = 0$ for $i \neq i^*$. Then, the last columns of $\boldsymbol{G}_k$ is $\check{\boldsymbol{h}}'_k$.

Let $\boldsymbol{G} = [\boldsymbol{G}_1 | \ldots | \boldsymbol{G}_{m_1} | \boldsymbol{O}]$ and $\boldsymbol{W} = [\boldsymbol{W}_1 | \ldots | \boldsymbol{W}_{m_1} | \boldsymbol{O}]$. The matrix $\boldsymbol{P} = [\check{\boldsymbol{p}}_1 \ \ldots \ \check{\boldsymbol{p}}_{m_1}]$ picks all columns $\check{\boldsymbol{h}}_1, \ldots, \check{\boldsymbol{h}}_{m_1}$ in $\boldsymbol{G}$ by setting $\check{\boldsymbol{p}}_j = \check{\boldsymbol{\imath}}_{\kappa j} \in R_{\mathbf{f}}^{m_2}$.

Length of $\boldsymbol{S}$: The norm of $\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})$ is $\max\{\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S}_1)\|, \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S}_2)\|\}$, where $\boldsymbol{S}_1 = [\boldsymbol{V}; \boldsymbol{P}]$ and $\boldsymbol{S}_2 = [\boldsymbol{D}; \boldsymbol{B}]$. For simplicity, we only consider the case where $\mathbf{f} = x^n + 1$. In the general case, the bound on $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})\|$ involves an extra $\mathrm{EF}(\mathbf{f}, 2)$ factor.

We have that $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{GB})\|^2 = \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{W})\|^2 \leq n$, since the entries of $\boldsymbol{W}$ are all 0 except the i^*-th polynomials $\mathbf{w}_{i^*,j}^{(k)}$ which are in $\{0,1\}^n$. As in the previous construction, we have $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{RB})\|^2 \leq 9nd$. Hence, we obtain that

$$\begin{aligned} \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S}_2)\|^2 &\leq \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{D})\|^2 + \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{B})\|^2 \leq \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{GB} + \boldsymbol{RB})\|^2 + \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{B})\|^2 \\ &\leq (3\sqrt{nd} + \sqrt{n})^2 + 5 \leq (4\sqrt{nd} + 3)^2. \end{aligned}$$

It is obvious that $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{P})\| \leq 1$. In addition, we have that $\|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{PR})\|^2 \leq nr$. Therefore,

$$\begin{aligned} \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S}_1)\|^2 &\leq \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{V})\|^2 + \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{P})\|^2 \leq \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{RP} - \boldsymbol{I})\|^2 + \|\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{P})\|^2 \\ &\leq (\sqrt{nd} + 1)^2 + 1 \leq (2\sqrt{nd} + 2)^2, \end{aligned}$$

which completes the proof.

10.6.4 Discussions

We left the problem to construct an analog of the Alwen–Peikert construction 3, which employs the rows of the Hadamard matrix to take a balance on the lower bound of m_2 and the norm of the Gram–Schmidt orthogonalized basis. The difficulty is finding the analog of the Hadamard matrix in $R_{\mathbf{f}}^{m_1}$ or the rows that have mutually orthogonality.

10.7 Ideal-Lattice-Based Collision-Resistant Preimage Sampleable Functions

By replacing the trapdoor-generation algorithm, we obtain the ideal-lattice-based collision-resistant PSFs ILPSF.

Scheme 10.7.1 (ILPSF [SSTX09])**.**

TrapGen(1^n)**:** It invokes ExtIdLattice1(1^n) (or ExtIdLattice2(1^n)) and obtains $(\breve{\boldsymbol{a}}, \boldsymbol{T})$. It outputs $(\breve{\boldsymbol{a}}, \boldsymbol{T})$, where $\breve{\boldsymbol{a}} \in R_{\mathbf{f},q}^m$ is statistically close to uniform and $\boldsymbol{T}' = \mathrm{Rot}_{\mathbf{f}}(\boldsymbol{T}) \subset \Lambda_q^{\perp}(\mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}}))$ is a good basis with $\|\tilde{\boldsymbol{T}}'\| \leq L$. The row vector $\breve{\boldsymbol{a}}$ defines the function $h_{\breve{\boldsymbol{a}}}(\cdot)$. This function is defined as $h_{\breve{\boldsymbol{a}}}(\boldsymbol{e}) = \mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}}) \cdot \boldsymbol{e} \bmod q$ with domain $D_n = \{\boldsymbol{e} \in \mathbb{Z}^{mn} : \|\boldsymbol{e}\|_\infty \leq s \log m\}$ and range $R_n = \mathbb{Z}_q^n$.

SampleDom(1^n)**:** The input distribution is $D_{\mathbb{Z}^{mn},s}$. It invokes SampleD($\boldsymbol{I}_{mn}, s, \mathbf{0}$) and outputs the obtained sample.

SamplePre($\breve{\boldsymbol{a}}, \boldsymbol{T}, s, \boldsymbol{u}$)**:** The algorithm samples from $h_{\breve{\boldsymbol{a}}}^{-1}(\boldsymbol{u})$ as follows: It generates $\boldsymbol{t} \in \mathbb{Z}^{mn}$ such that $\mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}})\boldsymbol{t} = \boldsymbol{u} \bmod q$ by standard algebra, samples $\boldsymbol{v} \leftarrow D_{\Lambda_q^{\perp}(\mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}})),s,-\boldsymbol{t}}$ by SampleD($\boldsymbol{T}', s, -\boldsymbol{t}$), and outputs $\boldsymbol{e} = \boldsymbol{t} + \boldsymbol{v}$.

In the following, we fix the polynomial $\mathbf{f} = x^n + 1$ with $n = 2^k \geq 32$. In addition, we fix q to be a prime with $q \equiv 3 \bmod 4$. We let denote $\Lambda_q^{\perp} = \Lambda_q^{\perp}(\mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}}))$ and $\Lambda_q = \Lambda_q(\mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}}))$.

We again start with several lemmas.

Instead of Lemma 10.5.2 we use the following lemma.

Lemma 10.7.2. *Let $m \geq 3$. Then, for all but an at most q^{-n} fraction of $\breve{\boldsymbol{a}}$, the columns of* $\mathrm{Rot}_{\mathbf{f}}(\breve{\boldsymbol{a}})$ *generates $\mathbb{Z}_q^n$.*

Proof. By the condition of $\mathbf{f}$ and $\mathbf{q}$, the row vector $\breve{\boldsymbol{a}}$ contains $\mathbf{a}_i \in R_{\mathbf{f},q}^*$ with probability at least $1 - (2q^{-n/2})^m \geq 1 - q^{-n}$. This completes the proof. □

Notice that we can apply Lemma 10.5.3 in our case. However, we cannot apply Lemma 10.5.4 directly in our case. Instead of the lemma, the following lemma ensures that for all but negligible fraction of $\breve{\boldsymbol{a}}$, we have $\lambda_1^{\infty}(\Lambda_q) \geq q/4$ and thus, for such $\breve{\boldsymbol{a}}$ and for any $\omega(\sqrt{\log mn})$ function, there exists a negligible function ϵ such that $\eta_\epsilon(\Lambda_q^{\perp}) \leq \omega(\sqrt{\log mn})$.

Lemma 10.7.3 (Lemma 5, [SSTX09]). *Let $m \geq 8 \log q$. Then for all but an at most q^{-n} fraction of $\check{\boldsymbol{a}} \in R_{\mathbf{f},q}^m$, we have $\lambda_1^\infty(\Lambda_q(\mathrm{Rot}_\mathbf{f}(\check{\boldsymbol{a}}))) \geq q/4$.*

For consistency, we include the proof, which is due to Stehlé and Steinfeld [SS09]. Before the proof, we should note the curious property of $\mathbf{f} = x^n + 1$.

Reciprocal polynomials: For a polynomial $\mathbf{a} = \sum_{i \in [n]} a_i x^{i-1}$, $\mathrm{Rot}_\mathbf{f}(\mathbf{a})$ is nega-cyclic matrix: Hence, we have the following relation on transpose operator.

$$\mathrm{Rot}_\mathbf{f}(\mathbf{a}) = \begin{pmatrix} a_1 & -a_n & \dots & -a_2 \\ a_2 & a_1 & \dots & -a_3 \\ \vdots & \vdots & \ddots & \vdots \\ a_n & a_{n-1} & \dots & a_1 \end{pmatrix} \text{ and } \mathrm{Rot}_\mathbf{f}(\mathbf{a})^T = \begin{pmatrix} a_1 & a_2 & \dots & a_n \\ -a_n & a_1 & \dots & a_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ -a_2 & -a_3 & \dots & a_1 \end{pmatrix}.$$

Let us consider $\mathbf{a}(1/x)$, that is, $\sum_{i \in [n]} a_i x^{-(i-1)}$. Since $x^{-(i-1)} = -x^i$ in $R_\mathbf{f}$, we have that

$$a_1 + a_2 x^{-1} + \cdots + a_n x^{-(n-1)} = a_1 - a_2 x^{n-1} - \cdots - a_n x^1 = a_1 - a_n x^1 - \cdots - a_2 x^{n-1}.$$

Now, we set $\mathrm{rec}(\mathbf{a}) = \mathbf{a}(1/x)$, a reciprocal polynomial of $\mathbf{a}$. Using this notion, we have that

$$\mathrm{Rot}_\mathbf{f}(\mathbf{a})^T = \mathrm{Rot}_\mathbf{f}(\mathrm{rec}(\mathbf{a})).$$

Obviously, the mapping rec is a bijection over $R_\mathbf{f}$ and $R_{\mathbf{f},q}$.

Returning to the proofs:

Proof. By our presupposition, we have that $\mathbf{f} = \mathbf{f}_1 \cdot \mathbf{f}_2$ over $\mathbb{Z}_q$ where $\mathbf{f}_i$ is irreducible in $\mathbb{Z}_q[x]$ and can be written $\mathbf{f}_i = x^{n/2} + t_i x^{n/4} - 1$ for some $t_i \in \mathbb{Z}_q$.

Let $\mathbf{s} \in R_{\mathbf{f},q}$ and $\boldsymbol{v} \in \mathbb{Z}_q^{mn}$. We want to bound the probability that $(\mathrm{Rot}_\mathbf{f}(\check{\boldsymbol{a}}))^T \cdot \mathbf{s} = \boldsymbol{v}$ when $\check{\boldsymbol{a}} \leftarrow R_{\mathbf{f},q}^m$. Since $\mathrm{Rot}_\mathbf{f}(\mathbf{a})^T = \mathrm{Rot}_\mathbf{f}(\mathrm{rec}(\mathbf{a}))$ and the mapping rec is a bijection over $R_{\mathbf{f},q}$, we instead bound the probability that $\check{\boldsymbol{a}} \otimes \mathbf{s} = \check{\boldsymbol{v}}$ for $\check{\boldsymbol{v}} \in R_{\mathbf{f},q}^m$. Let us define the map $\phi_\mathbf{s}$ that maps $\mathbf{a}$ to $\mathbf{a} \otimes \mathbf{s}$. The probability is $\prod_{j \in [m]} \Pr_{\mathbf{a}_j \leftarrow R_{\mathbf{f},q}}[\phi_\mathbf{s}(\mathbf{a}_j) = \mathbf{v}_j]$.

The case where $\mathbf{s}$ *and* $\mathbf{f}$ *are coprime:* Since $\phi_\mathbf{s}$ is a bijection in this case, we have that $\Pr_{\mathbf{a}_j \leftarrow R_{\mathbf{f},q}}[\phi_\mathbf{s}(\mathbf{a}_j) = \mathbf{v}_j]$ is q^{-n}.

The case where $\mathbf{s}$ *and* $\mathbf{f}$ *are not coprime:* In this case, we have $\mathbf{s} = \mathbf{f}_i \mathbf{s}'$ for some $i \in \{1, 2\}$ and $\mathbf{s}' \in \mathbb{Z}_q[x]$ of degree smaller than $n/2$. If $\mathbf{v}_j$ is not of the form $\mathbf{f}_i \mathbf{v}'_j$ for some $\mathbf{v}'_j$ of degree smaller than $n/2$, then $\Pr_{\mathbf{a}_j}[\phi_\mathbf{s}(\mathbf{a}_j) = \mathbf{v}_j] = 0$. Otherwise, since the kernel of $\phi_\mathbf{s}$ is of cardinality $q^{n/2}$, we have $\Pr_{\mathbf{a}_j}[\phi_\mathbf{s}(\mathbf{a}_j) = \mathbf{v}_j] = q^{-n/2}$.

Taking the union bound over all non-zero polynomials $\mathbf{s} \in R_{\mathbf{f},q}$ and the vectors $\check{\boldsymbol{v}} \in R_{\mathbf{f},q}^m$ such that $\|\check{\boldsymbol{v}}\|_\infty < q/4$, the probability that we have $\lambda_1^\infty(\Lambda_q(\mathrm{Rot}_\mathbf{f}(\check{\boldsymbol{a}}))) < q/4$ is upper bounded by

$$\sum_{\substack{\mathbf{s} \in R_{\mathbf{f},q} \\ \gcd(\mathbf{s},\mathbf{f})=1}} \sum_{\substack{\check{\boldsymbol{v}} \in R_{\mathbf{f},q}^m \\ \|\check{\boldsymbol{v}}\|_\infty < q/4}} \prod_{j \in [m]} \Pr_{\mathbf{a}}[\phi_\mathbf{s}(\mathbf{a}) = \mathbf{v}_j] + 2 \sum_{\substack{\mathbf{s} \in R_{\mathbf{f},q} \\ \mathbf{f}_1 | \mathbf{s}}} \sum_{\substack{\check{\boldsymbol{v}} \in R_{\mathbf{f},q}^m \\ \|\check{\boldsymbol{v}}\|_\infty < q/4}} \prod_{j \in [m]} \Pr_{\mathbf{a}}[\phi_\mathbf{s}(\mathbf{a}) = \mathbf{v}_j].$$

The first term is upper bounded by $q^n(q/2)^m q^{-mn} = 2^{-m} q^{-mn+m+n}$. Let N be the number of $\mathbf{v} \in R_{\mathbf{f},q}$ such that $\|\mathbf{v}\|_\infty < q/4$ and $\mathbf{v} = \mathbf{f}_1\mathbf{v}'$ for some $\mathbf{v}'$. Thanks to the shape of $\mathbf{f}_1 = x^{n/2} - t_1 x^{n/4} + 1$, the latter conditions imply that $\|\mathbf{v}''\|_\infty < q/4$ where $\mathbf{v}'' \in \mathbb{Z}_q[x]$ is the vector made of the $n/4$ lower degree coefficients of $\mathbf{v}'$. Hence, we have that $N \leq q^{n/2}/2^{n/4}$. Therefore, the second term is at most $2q^{n/2}N^m q^{-mn/2} = 2q^{n/2}2^{-nm/4}$.

This argument shows that the probability we have the short vector $\boldsymbol{v}$ corresponding to $\check{\boldsymbol{v}}$ in $\Lambda_q(\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}}))$ is at most

$$2^{-m} q^{-mn+m+n} + 2q^{n/2}2^{-nm/4},$$

which is negligible when $m = 2(1+\delta)\log q$. In particular, if we set $m = 8\log q$, the probability is at most

$$q^{-n-(mn+8-m-2n)} + 2q^{n/2}q^{-2n} \leq q^{-n}.$$

□

We can show the following corollary.

Corollary 10.7.4. *Let $m \geq 8\log q$. Then for all but an at most $2q^{-n}$ fraction of $\check{\boldsymbol{a}} \in R_{\mathbf{f},q}$ and for any $s \geq \omega(\sqrt{\log mn})$, the distribution of the syndrome $\mathbf{u} = \check{\boldsymbol{a}}\check{\boldsymbol{e}} \bmod q$ is statistically close to uniform over $R_{\mathbf{f},q}$, where $\check{\boldsymbol{e}} = \boldsymbol{e} \leftarrow D_{\mathbb{Z}^{mn},s}$.*

Proof. By Lemma 10.7.2 and Lemma 10.7.3, for all but a $2q^{-n}$ fraction of all $\check{\boldsymbol{a}}$, the columns of $\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})$ generate $\mathbb{Z}_q^n$ and $s \geq \eta_\epsilon(\Lambda_q^\perp)$ for some negligible function $\epsilon(mn)$. Now by Lemma 10.5.3, the distribution of $\mathbf{u} = \check{\boldsymbol{a}}\check{\boldsymbol{e}} = \mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}}) \cdot \boldsymbol{e} \bmod q$ is statistically close to uniform over $R_{\mathbf{f},q}$. □

By using this lemma, we can proof the security of ILPSF.

Theorem 10.7.5 ([SSTX09]). *Let $\mathbf{f} = x^n + 1$ and $n = 2^k \geq 32$. Let m and q be integers with q prime, $q \equiv 3 \bmod 4$, and $m \geq 41\log q$. Then, the above scheme ILPSF is collision resistant if $\mathbf{f}$-$\mathrm{SIS}_{q,m,2s\sqrt{mn}}$ is hard.*

Proof. We note that $s \geq L \cdot \omega(\sqrt{\log m}) \geq \eta_\epsilon(\Lambda_q^\perp)$ for some negligible $\epsilon(n)$ by Lemma 2.1.7 since $L \geq \|\tilde{\boldsymbol{T}}'\| \geq \tilde{bl}(\Lambda_q^\perp)$.

Next, a sample $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^{mn},s}$ falls into D_n except with negligible probability by Lemma 2.1.9. Furthermore, for all but a $2q^{-n}$ fraction of $\boldsymbol{A}$, $h_{\boldsymbol{A}}(\boldsymbol{e})$ is statistically close to uniform over $R_n = \mathbb{Z}_q^n$ by Corollary 10.7.4.

The preimage sampleable property follows from $s \geq \|\tilde{\boldsymbol{T}}\| \cdot \omega(\sqrt{\log m})$, Lemma 10.5.3, and the correctness of SampleD (Theorem 10.4.7); The samples from a distribution is statistically close to $D_{\Lambda_q^\perp,s,-\boldsymbol{t}}$ and the conditional distribution of $\boldsymbol{e} \leftarrow D_{\mathbb{Z}^{mn},s}$ given $\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})\boldsymbol{e} \equiv \boldsymbol{u} \pmod q$ is exactly $\boldsymbol{t} + D_{\Lambda_q^\perp,s,-\boldsymbol{t}}$.

The collision resistance property immediately follows from the hardness of $\mathbf{f}$-$\mathrm{SIS}^\infty_{q,m,2s\log mn}$ or $\mathbf{f}$-$\mathrm{SIS}_{q,m,2s\sqrt{mn}}$

The preimage min-entropy is at least $mn-1$. This follows the fact that the preimages are distributed according to $\boldsymbol{t} + D_{\Lambda^\perp,s,-\boldsymbol{t}}$ and the min-entropy of $D_{\Lambda^\perp,s,-\boldsymbol{t}}$ is at least $mn-1$ (see Lemma 2.1.14). □

10.8 On “Bonsai” Notions

Peikert [Pei09b] compared the generations of the random lattice $\boldsymbol{A}$ to controlling growths in Bonsai. Let us figure out what is a *bonsai tree*. Imagine the binary tree $\{0,1\}^l$ and the path μ. Let $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{A}_i^{(b)} \in \mathbb{Z}_q^{n\times m}$ for $i \in [l]$ and $b \in \{0,1\}$. Then, for any $\mu \in \{0,1\}^l$, we define $\boldsymbol{A}_\mu = [\boldsymbol{A}_0|\boldsymbol{A}_1^{(\mu_1)}|\dots|\boldsymbol{A}_l^{(\mu_l)}]$. This construction indicates a *hierarchy of trapdoor functions*.

The legitimate user has a trapdoor $\boldsymbol{T}_0$ of $\boldsymbol{A}_0$ and generate random $\boldsymbol{A}_i^{(b)}$, which is *undirected growth*. It then has, for any $\mu \in \{0,1\}^l$, a trapdoor $\boldsymbol{T}_\mu$ for $\boldsymbol{A}_\mu$ by *extending control*.

The simulator crucially uses a *directed growth*. From $\boldsymbol{A}_0$, it can makes $\boldsymbol{A}_1$ with a trapdoor of $[\boldsymbol{A}_0|\boldsymbol{A}_1]$.

In addition, we can delegate the basis by a *randomized control* (see also [CHK09]). If one knows a trapdoor $\boldsymbol{T}$ of $\boldsymbol{A}$, one can generates a new trapdoor $\boldsymbol{S}$ of $\boldsymbol{A}$ with a slight loss.

These techniques will be exploited in digital signature (Chapter 11), public-key encryption (Chapter 12), and identity-based encryption (Chapter 14).

In the following, m denotes the sum of m_1 and m_2.

10.8.1 Undirected Growth

Let $\boldsymbol{A}_1 \in \mathbb{Z}_q^{n\times m_1}$ and let $\boldsymbol{A}_2 \in \mathbb{Z}_q^{n\times m_2}$. Let us define $\boldsymbol{A} = [\boldsymbol{A}_1|\boldsymbol{A}_2]$. It is obvious that $\Lambda_q^\perp(\boldsymbol{A})$ is a higher-dimensional supper-lattice of $\Lambda_q^\perp(\boldsymbol{A})$, since, for any $\boldsymbol{v}_1 \in \Lambda_q^\perp(\boldsymbol{A}_1)$, the vector $\boldsymbol{v} = \boldsymbol{v}_1 \circ \boldsymbol{0}$ is in $\Lambda_q^\perp(\boldsymbol{A})$. Undirected growth is done by concatenating fresh random matrix $\boldsymbol{A}_2$ onto a given $\boldsymbol{A}_1$.

10.8.2 Controlled Growth

This was already done by Alwen and Peikert. An arborist can generate a lattice $\Lambda_q^\perp(\boldsymbol{A})$ with a short basis of it from $\boldsymbol{A}_1$ as in the constructions (see Sections 10.3.2, 10.3.3, and 10.3.4).

10.8.3 Extending Control

If an arborist knows a trapdoor $\boldsymbol{T}$ of $\boldsymbol{A}_1$, then he also knows a trapdoor $\boldsymbol{S}$ of $\boldsymbol{A} = [\boldsymbol{A}_1|\boldsymbol{A}_2]$. We suppose that q is a prime.

Let us consider the following deterministic algorithm $\mathsf{ExtBasis}(\boldsymbol{T}, \boldsymbol{A})$ which will outputs $\boldsymbol{S}$;

- For $j = 1, \dots, m_1$, let $\boldsymbol{s}_i = \boldsymbol{t}_i \circ \boldsymbol{0} \in \mathbb{Z}^m$.
- For $j = 1, \dots, m_2$, let $\boldsymbol{b}_i \in \mathbb{Z}^m$ be an arbitrary integer solution to the equation $\boldsymbol{A}_1\boldsymbol{b}_i \equiv -\boldsymbol{a}_i^{(2)} \pmod{q}$, where $\boldsymbol{A}_2 = [\boldsymbol{a}_1^{(2)}, \dots, \boldsymbol{a}_{m_2}^{(2)}]$. Let $\boldsymbol{s}_{m_1+i} = \boldsymbol{b}_i \circ \boldsymbol{i}_i \in \mathbb{Z}^m$.

It is easy to show the following lemma, which states the matrix $\boldsymbol{S}$ inherits the quality of $\boldsymbol{T}$.

Lemma 10.8.1 (Lemma 3.2, [Pei09b])**.** *The algorithm runs in polynomial-time and outputs a basis $\boldsymbol{S}$ of $\Lambda_q^{\perp}(\boldsymbol{A})$ such that $\|\tilde{\boldsymbol{S}}\| = \|\tilde{\boldsymbol{T}}\|$.*

10.8.4 Randomizing Control

We finally review how to randomize the basis of a superlattice $\Lambda_q^{\perp}(\boldsymbol{A})$. This control also appears in Cash, Hofheinz, and Kiltz [CHK09] with a name, delegation of a basis.

Let us consider the following probabilistic algorithm $\mathsf{RandBasis}(\boldsymbol{T}, s)$ which, given a basis $\boldsymbol{T}$ of some m-dimensional lattice Λ and a parameter $s \geq \|\tilde{\boldsymbol{T}}\| \cdot \omega(\sqrt{\log n})$, outputs a new basis $\boldsymbol{S}$ of Λ;

1. For $i = 1, \dots, m$,
   (a) Generate $\boldsymbol{v} \leftarrow \mathsf{SampleD}(\boldsymbol{T}, s)$. If $\boldsymbol{v}$ is linearly independent of $\{\boldsymbol{v}_1, \dots, \boldsymbol{v}_{i-1}\}$, then let $\boldsymbol{v}_i = \boldsymbol{v}$ and increment i. Otherwise, repeat this step.
2. Let $\boldsymbol{T}'$ be an HNF of $\boldsymbol{T}$. Output $\boldsymbol{S} \leftarrow \mathsf{MGReduce}(\boldsymbol{T}', \boldsymbol{V})$.

It is easy to verify that Step 1 takes at most $O(m^2)$ times with overwhelming probability. On the quality, $\|\tilde{\boldsymbol{S}}\| \leq s \cdot \sqrt{m}$ with overwhelming probability, because $\boldsymbol{v}$ distributes according to the distribution statistically close to $D_{\Lambda,s}$ by Theorem 10.4.7 and the norm bound in Theorem 2.1.9.

10.9 On “Miniature Bonsai” Notions

Here, we apply the above techniques to the ideal-lattice-based constructions. In the following, m denotes the sum of m_1 and m_2.

10.9.1 Undirected Growth

Let $\check{\boldsymbol{a}}_1 \in R_{\mathbf{f},q}^{m_1}$ and let $\check{\boldsymbol{a}}_2 \in R_{\mathbf{f},q}^{m_2}$. Let us define $\check{\boldsymbol{a}} = \check{\boldsymbol{a}}_1 \circ \check{\boldsymbol{a}}_2$. As in the previous section, it is obvious that $M_q^{\perp}(\check{\boldsymbol{a}})$ is a higher-dimensional supper-module of $M_q^{\perp}(\check{\boldsymbol{a}}_1)$, since, for any $\check{\boldsymbol{e}}_1 \in M_q^{\perp}(\check{\boldsymbol{a}}_1)$, the vector $\check{\boldsymbol{e}} = \check{\boldsymbol{e}}_1 \circ \check{\boldsymbol{0}}$ is in $M_q^{\perp}(\boldsymbol{A})$. Undirected growth is done by concatenating fresh random vector $\check{\boldsymbol{a}}_2$ onto a given $\check{\boldsymbol{a}}_1$.

10.9.2 Controlled Growth

An arborist can generate a module $M_q^{\perp}(\check{\boldsymbol{a}})$ with a short basis $\boldsymbol{T}$ of it from $\check{\boldsymbol{a}}_1$ as in the SSTX constructions (see Section 10.6).

10.9.3 Extending Control

If an arborist knows a trapdoor $\boldsymbol{T}$ of $\check{\boldsymbol{a}}_1$, then he also knows a trapdoor $\boldsymbol{S}$ of $\check{\boldsymbol{a}} = \check{\boldsymbol{a}}_1 \circ \check{\boldsymbol{a}}_2$.

Let us consider the following deterministic algorithm $\mathsf{ExtIdBasis}(\boldsymbol{T}, \boldsymbol{A})$ which will outputs $\boldsymbol{S} = [\check{\boldsymbol{s}}_1, \ldots, \check{\boldsymbol{s}}_m]$;

- For $j = 1, \ldots, m_1$, let $\check{\boldsymbol{s}}_j = \check{\boldsymbol{t}}_j \circ \check{\boldsymbol{0}} \in R_{\mathbf{f}}^m$.
- For $j = 1, \ldots, m_2$, let $\check{\boldsymbol{b}}_j \in R_{\mathbf{f}}^{m_1}$ be an arbitrary integer solution to the equation $\check{\boldsymbol{a}}_1 \boldsymbol{b}_j \equiv -\mathbf{a}_j^{(2)} \pmod{q}$, where $\check{\boldsymbol{a}}_2 = [\mathbf{a}_1^{(2)}, \ldots, \mathbf{a}_{m_2}^{(2)}]$. Let $\check{\boldsymbol{s}}_{m_1+j} = \check{\boldsymbol{b}}_j \circ \check{\boldsymbol{\iota}}_j \in \mathbb{Z}^m$.

It is easy to show the analogue of Lemma 10.8.1 which states the matrix $\boldsymbol{S}$ inherits the quality of $\boldsymbol{T}$.

Lemma 10.9.1. *The algorithm runs in polynomial-time and outputs a basis* $\boldsymbol{S}$ *of* $M_q^{\perp}(\check{\boldsymbol{a}})$ *such that* $\|\widetilde{\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{S})}\| = \|\widetilde{\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{T})}\|$.

10.9.4 Randomizing Control

We can use $\mathsf{RandBasis}$ appeared in Section 10.8. Hence, we omit the details.

10.10 An Application: Trapdoor Hash Functions

Introduction: Trapdoor commitments (or trapdoor hash functions) appeared first in Brassard, Chaum, and Crépeau [BCC88], under the name "chameleon blobs", in the context of zero-knowledge proofs and arguments. They are underlying primitives to construct complex cryptographic schemes and have many applications, zero-knowledge proofs, arguments, signatures, universally composable commitments. See Fischlin's thesis [Fis01] for the details of trapdoor commitment (which is a generalized notion of trapdoor hash family).

Here, we intend to discuss non-interactive one, a trapdoor hash family as known as "chameleon hash functions" [KR00]. We mainly adopt the definition of trapdoor hash functions by Shamir and Tauman (Kalai) in [ST01], however, their definition depends on the number-theoretic assumptions: they require uniformity of hash functions.

Let us confirm the definitions in [KR00, ST01]. Roughly speaking, the scheme consists of a triple of polynomial-time algorithm, $(\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{TrapCol})$; The generation algorithm, given the security parameter 1^n, outputs (a, t), a pair of an index of hash function, which defines a hash function $h_a : M_{n,a} \times W_{n,a} \to R_{n,a}$, and a trapdoor corresponding to a; The evaluation algorithm Eval, given a, $m \in M_{n,a}$, and $r \in W_{n,a}$, computes $d = h_a(m, r)$; The trapdoor collision algorithm $\mathsf{TrapCol}$, given t, two distinct messages $m_1 \neq m_2 \in M_{n,a}$, and $r_1 \in R_{n,a}$, outputs $r_2 \in R_{n,a}$ such that $h_a(m_1, r_1) = h_a(m_2, r_2)$; As ordinal hash functions, it is required to be collision resistant: any polynomial-time adversary cannot, given an index a, outputs two distinct message $m_1 \neq m_2$ and two strings r_1 and r_2 such that $h_a(m_1, r_1) = h_a(m_2, r_2)$, where the probability is taken over the choice of $(a, t) \leftarrow \mathsf{TrapGen}(1^n)$ and the randomness of the adversary. The problem is in the definition of uniformity. In [KR00] and [ST01], their requirements are as follows:

By Krawczyk and Rabin [KR00]: For any two distinct messages $m_1 \neq m_2 \in M_{n,a}$, the two distributions $h_a(m_1, r_1)$ and $h_a(m_2, r_2)$ are computationally identical if r_1 and r_2 are chosen uniformly at random from $R_{n,a}$.

By Shamir and Tauman [ST01]: If r_1 is uniformly distributed in $W_{n,a}$, then the distribution of r_2 output by the algorithm $\mathsf{TrapCol}$ is computationally indistinguishable from uniform in $W_{n,a}$.

These uniformity definitions are violated if we choose r_1 from another distribution over $W_{n,a}$ rather than the uniform distribution. Hence, we here give more generalized one, which is very similar to the definition of preimage sampleable functions in Section 10.2.

10.10.1 Definitions

Model of Trapdoor Hash Functions

In order to introduce the other distribution over $W_{n,a}$, we add the algorithm $\mathsf{SampleDom}$ to the scheme. Let $\mathsf{THash} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SampleDom}, \mathsf{TrapCol})$ over a message space M_n, a randomness space W_n, and a value space R_n be a trapdoor hash scheme. Notation of the algorithms is below:

$\mathsf{TrapGen}(1^n)$**:** A key-generation algorithm, given the security parameter 1^n, outputs a pair of an index of a hash function and a trapdoor (a, t). An index a defines the hash function $h_a : M_n \times W_n \to R_n$.

$\mathsf{Eval}(a, msg, r)$**:** An evaluation algorithm, given a, a message $msg \in M_n$ and a randomness $r \in W_n$, outputs a digest $d = h_a(msg, r) \in R_n$.

$\mathsf{SampleDom}(1^n)$**:** A domain sampling algorithm, given the security parameter 1^n, samples $r \in W_n$ from some distribution over W_n.

$\mathsf{TrapCol}(a, t, msg_0, r_0, msg_1)$**:** A trapdoor collision algorithm, given a, t corresponding to a, $msg_0 \in M_n$, $r_0 \in R_n$, and $msg_1 \in M_n$, outputs $r_1 \in R_n$.

Security Notions

The collision resistance is defined as that in usual hash functions (see Section 4.1). The hiding property is given below in order to generalize the uniformity in the previous definitions: If $\mathsf{SampleDom}$ samples from the uniform distribution over R_n, the definition is just the uniformity.

To define the security notion, consider the experiments $\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{THash},\mathcal{A}}(n)$ and $\mathbf{Exp}^{\mathrm{hide}}_{\mathsf{THash},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{THash},\mathcal{A}}(n)$**:**

Setup Phase: The challenger C runs $\mathsf{TrapGen}(1^n)$ and obtains (a, t). The adversary $\mathcal{A}$ is given the security parameter 1^n and the parameters a.

Challenge Phase: The adversary outputs (msg, r), and (msg', r'). If $msg, msg' \in M_n$, $r, r' \in W_n$, $(msg, r) \neq (msg', r')$, and $\mathsf{Eval}(a, msg, r) = \mathsf{Eval}(a, msg', r')$ then C returns 1. Otherwise, it returns 0.

Experiment $\mathbf{Exp}^{\mathrm{hide}}_{\mathsf{THash},\mathcal{A}}(n)$:

Setup Phase: The challenger C runs $\mathsf{TrapGen}(1^n)$ and obtains (a, t). The adversary $\mathcal{A}$ is given the security parameter 1^n and the parameters *param*.

Challenge Phase: The adversary outputs msg_0 and msg_1. If (1) $msg_0, msg_1 \in M_n$ and (2) $msg_0 \neq msg_1$, the challenger flips a fair coin $b \leftarrow \{0, 1\}$. If $b = 0$, C generates $r_0 \leftarrow \mathsf{SampleDom}(1^n)$. If $b = 1$, C generates $r_0 \leftarrow \mathsf{SampleDom}(1^n)$ and obtains $r_1 \leftarrow \mathsf{TrapCol}(a, t, msg_0, r_0, msg_1)$. C provides (msg_b, r_b) to the adversary $\mathcal{A}$. If the above two checks are not passed, C returns 0 and halts.

Decision Phase: Finally, the adversary outputs its decision b'. If $b = b'$ the challenger returns 1, otherwise 0.

Definition 10.10.1 (Collision resistance)**.** Let THash be a trapdoor hash scheme. Let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{THash},\mathcal{A}}(n) := \Pr\left[\mathbf{Exp}^{\mathrm{cr}}_{\mathsf{THash},\mathcal{A}}(n) = 1\right].$$

We say a trapdoor hash scheme THash is computationally binding if $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{THash},\mathcal{A}}(n)$ is negligible in n for any polynomial-time adversary $\mathcal{A}$.

We treat the uniformity as the property of the output distribution of h_a.

Definition 10.10.2 (Uniformity)**.** Consider a trapdoor hash scheme THash. We say THash has the (statistical) uniformity if if any messages $msg \in M_n$, $\Delta((a, d), (a, u)) \leq \mathsf{negl}(n)$ for some negligible function $\mathsf{negl}(n)$, where $(a, t) \leftarrow \mathsf{TrapGen}(1^n)$, $r \leftarrow \mathsf{SampleDom}(1^n)$, $d \leftarrow \mathsf{Eval}(msg, r)$, and $u \leftarrow R_n$.

The property that anyone cannot distinguish the pair of a message and a random string and the pair output by $\mathsf{TrapCol}$ is now named as the hiding property.

Definition 10.10.3 (Hiding)**.** Consider a trapdoor hash scheme THash. Let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{THash},\mathcal{A}}(n) := \left|\Pr\left[\mathbf{Exp}^{\mathrm{hide}}_{\mathsf{THash},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say THash is computationally hiding if for any polynomial-time adversary $\mathcal{A}$, $\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{THash},\mathcal{A}}(n)$ is negligible in n.

10.10.2 Constructions

Fujisaki [Eii08] and Peikert [Pei09b] pointed out the construction of trapdoor hash schemes based on lattice problems. These are based on LPSF with the flavor of LNIC.

In LNIC, an index of the hash function is $\boldsymbol{A} = [\boldsymbol{A}'|\boldsymbol{A}''] \in \mathbb{Z}_q^{n\times(m'+m'')}$. The commitment $\boldsymbol{u} = \boldsymbol{A}(\boldsymbol{m} \circ \boldsymbol{r}) = \boldsymbol{A}'\boldsymbol{m} + \boldsymbol{A}''\boldsymbol{r}$. What happen if we have the trapdoor of $\boldsymbol{A}''$? For any message $\boldsymbol{m}$ and a committed value $\boldsymbol{u}$, we can sample $\boldsymbol{r}$ such that $\boldsymbol{A}''\boldsymbol{r} = \boldsymbol{u} - \boldsymbol{A}'\boldsymbol{m}$.

Descriptions

Scheme 10.10.4 (LTHash)**.**

TrapGen(1^n)**:** The key-generation algorithm, given the security parameter 1^n, obtains $(\boldsymbol{A}', \boldsymbol{T}') \leftarrow$ LPSF.TrapGen(1^n) with the parameter m. Next, it generates a random matrix $\boldsymbol{A}'' \leftarrow \mathbb{Z}_q^{n\times m}$. An index $\boldsymbol{A} = [\boldsymbol{A}''|\boldsymbol{A}']$ defines the hash function $h_{\boldsymbol{A}} : \{0,1\}^m \times D_n \to \mathbb{Z}_q^n$.

Eval($\boldsymbol{A}, \boldsymbol{w}, r$)**:** The evaluation algorithm, given $\boldsymbol{A}$, a message $msg = \boldsymbol{w} \in \{0,1\}^m$ and a randomness $\boldsymbol{r} \in D_n$, outputs a digest $\boldsymbol{u} = h_{\boldsymbol{A}}(\boldsymbol{w} \circ \boldsymbol{r}) \in \mathbb{Z}_q^n$.

SampleDom(1^n)**:** The domain sampling algorithm is the same as LPSF.SampleDom($\cdot$). The distribution is $D_{\mathbb{Z}^m,s}$.

TrapCol($\boldsymbol{A}, \boldsymbol{T}, \boldsymbol{w}_0, \boldsymbol{r}_0, \boldsymbol{w}_1$)**:** First, it computes a digest $\boldsymbol{u} = h_{\boldsymbol{A}}(\boldsymbol{w}_0 \circ \boldsymbol{r}_0) = \boldsymbol{A}''\boldsymbol{w}_0 + \boldsymbol{A}'\boldsymbol{r}_0$. Then, it computes a half of digest $\boldsymbol{u}' = \boldsymbol{A}''\boldsymbol{w}_1$. Since, $\boldsymbol{A}'\boldsymbol{r}_1 = \boldsymbol{u}' - \boldsymbol{u}$, it obtains $\boldsymbol{r}_1 \leftarrow$ LPSF.SamplePre($\boldsymbol{A}', \boldsymbol{T}', s, \boldsymbol{u}' - \boldsymbol{u}$). Then, it outputs $\boldsymbol{r}_1$.

Security Proofs

The security proofs are straightforward. The collision resistance and the uniformity follow from these of LPSF. The hiding property also follows statistical one of LPSF. Hence, we omit them.

Extension

We again extend the domain of messages. As already noted in [KR00], the combination of hash functions and trapdoor hash functions yields this; the new trapdoor hash functions are in the form $h'_a(msg, r) = h_a(H(msg), r)$, where h_a is a trapdoor hash function and H is a hash function. We note that our method in Section 5.3.1 also extend the domain in our case.

An Ideal-Lattice-Based Construction

By simple argument, we can construct ILThash from ILPSF. We omit the details, since they are very similar to LTHash from LPSF.

Remark

We finally note that Kurosawa and Heng [KH08] showed the conversion from a trapdoor hash family to an ID scheme. Their conversion yields the following ID scheme. The key pair is $(a, t) \leftarrow$ TrapGen(1^n). The protocol is defined as follows:

(1) the prover chooses $m \leftarrow M_n$, generates a sample $r \leftarrow \mathsf{SampleDom}(1^n)$, and commits $y = h_a(m, r) = \mathsf{Eval}(a, m, r)$, (2) the verifier sends a random challenge $c \leftarrow M_n$, (3) the prover computes $z \leftarrow \mathsf{TrapCol}(a, t, m, r, c)$ and sends it, and (4) the verifier checks that $z \in R_n$ and $y = h_a(c, z)$.

Applying this conversion to our LTHash, we obtain a passively-secure ID scheme based on the average-case hardness of $\mathrm{SIS}_{q,m,\beta}$ for $\beta = \tilde{O}(n)$ (thus, the worst-case hardness of $\mathrm{SIVP}_{\tilde{O}(n^{1.5})}$). The protocol is quite efficient since the communication cost is $\tilde{O}(n)$.

11
Signature

Organization: This chapter includes the signature schemes based on lattice problems. We give the definitions of signature in Section 11.1. Section 11.2 summarizes the general conversions from several primitives to secure signature schemes. In Section 11.3, we review the Gentry–Peikert–Vaikuntanathan signature scheme. Section 11.4 gives a description of the ideal-lattice version of it by Stehlé et al.. Section 11.5 reviews the ideal-lattice-based one-time signature by Lyubashevsky and Micciancio. Section 11.6 reviews the obtained signatures from Lyubashevsky's ID scheme. In Section 11.7 gives a brief review of a signature scheme proposed by Peikert very recently.

11.1 Definitions

11.1.1 Model of Signature Schemes

A signature scheme is a quadruplet of algorithms $\mathsf{SIG} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$.

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs public parameters *param*.

$\mathsf{KeyGen}(param)$**:** A key-generation algorithm, given *param*, outputs a pair of a verification key and a signing key (vk, sk).

$\mathsf{Sign}(param, sk, msg)$**:** A signing algorithm, given *param*, *sk*, and a message *msg*, outputs a signature σ.

$\mathsf{Ver}(param, vk, msg, \sigma)$**:** A verification algorithm, given *param*, *vk*, *msg*, and σ, returns 0 (reject) or 1 (accept).

Correctness: We require a correctness condition that for any *msg*, it holds that $\mathsf{Ver}(param, vk, msg, \sigma) = 1$ with overwhelming probability for correctly generated *param*, (vk, sk), and σ. Formally, we require that for any *msg*,

$$\Pr\left[dec = 1 : \begin{array}{l} param \leftarrow \mathsf{Setup}(1^n); \\ (vk, sk) \leftarrow \mathsf{KeyGen}(param); \\ \sigma \leftarrow \mathsf{Sign}(param, sk, msg); \\ dec \leftarrow \mathsf{Ver}(param, vk, msg, \sigma); \end{array} \right] = 1 - \mathsf{negl}(n).$$

11.1.2 Security Notions

The required security is basically that any polynomial-time adversary cannot output a valid signature even if it can choose a message adversely. The notion is called as *existential unforgeability*. There are several attacks and we describe the formal definitions as follows:

One-time security means any polynomial-time adversary cannot output a valid signature even if it is provided a signature of any message, which can be chosen adversary, made by a valid signer. Consider the experiment $\mathbf{Exp}^{\text{ot-cma}}_{\mathsf{SIG},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\text{ot-cma}}_{\mathsf{SIG},\mathcal{A}}(n)$:

Setup Phase: The challenger C takes a security parameter 1^n. The challenger runs the algorithm Setup, and obtains parameters *param*. Next, it obtains $(vk, sk) \leftarrow \mathsf{KeyGen}(param)$. C gives 1^n, *param*, and *vk* to the adversary.

Learning Phase: The adversary queries to the oracle SIGN at most once.

- The oracle SIGN receives a message *msg*. It returns $\sigma \leftarrow \mathsf{Sign}(param, sk, msg)$ to the adversary.

Challenge Phase: The adversary $\mathcal{A}$ outputs a message msg^* and a forged signature σ^*. If $msg^* \neq msg$ and $\mathsf{Ver}(param, vk, msg^*, \sigma^*) = 1$, then C outputs 1. Otherwise, it outputs 0.

Definition 11.1.1 (One-time security)**.** Let SIG be a signature scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ot-cma}}_{\mathsf{SIG},\mathcal{A}}(n) = \Pr\left[\mathbf{Exp}^{\text{ot-cma}}_{\mathsf{SIG},\mathcal{A}}(n) = 1\right].$$

We say that SIG is one-time secure if $\mathbf{Adv}^{\text{ot-cma}}_{\mathsf{SIG},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

Furthermore, we say that SIG is strongly one-time secure if $\mathbf{Adv}^{\text{ot-cma}}_{\mathsf{SIG},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$, where we replace the check $msg^* \neq msg$ with $(msg^*, \sigma^*) \neq (msg, \sigma)$ in the challenge phase of the experiment.

Existential unforgeability under weak chosen message attacks means any polynomial-time adversary cannot output a valid signature of an unsigned message even if it is given signatures of chosen message before it is given the verifying key. Consider the experiment $\mathbf{Exp}^{\text{euf-wcma}}_{\mathsf{SIG},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\text{euf-wcma}}_{\mathsf{SIG},\mathcal{A}}(n)$**:**

Initiating Phase: The challenger C takes a security parameter 1^n. C gives $\mathcal{A}$ the security parameter 1^n. C receives $(msg_1, \dots, msg_l)$ from $\mathcal{A}$.

Setup Phase: The challenger runs the algorithm Setup, and obtains parameters *param*. Next, it obtains $(vk, sk) \leftarrow \mathsf{KeyGen}(param)$.

Learning Phase: The challenger makes signatures σ_i for the message msg_i using sk. Then, it feeds vk and $\sigma_1, \dots, \sigma_l$ to the adversary.

Challenge Phase: The adversary $\mathcal{A}$ outputs a message msg^* and a forged signature σ^*. If $msg^* \neq msg_i$ for any i and $\mathsf{Ver}(param, vk, msg^*, \sigma^*) = 1$, then C outputs 1. Otherwise, it outputs 0.

Definition 11.1.2 (EUF-wCMA security)**.** Let SIG be a signature scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{euf-wcma}}_{\mathsf{SIG},\mathcal{A}}(n) = \Pr\left[\mathbf{Exp}^{\text{euf-wcma}}_{\mathsf{SIG},\mathcal{A}}(n) = 1\right].$$

We say that SIG is existentially unforgeable under weak chosen message attacks if $\mathbf{Adv}^{\text{euf-wcma}}_{\mathsf{SIG},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

Existential unforgeability under chosen message attacks (EUF-CMA) means any polynomial-time adversary cannot output a valid signature even if it is provided a signature of any message, which can be chosen adversary, made by a valid signer. In strong EUF-CMA (sEUF-CMA), the adversary wins if the output message is already signed (we need the output signature is not equal to the signature output by the legitimate signer).

Consider the experiment $\mathbf{Exp}^{\text{goal-cma}}_{\mathsf{SIG},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$, where goal $\in \{\text{euf}, \text{seuf}\}$.

Experiment $\mathbf{Exp}^{\text{goal-cma}}_{\mathsf{SIG},\mathcal{A}}(n)$**:**

Setup Phase: The challenger C takes a security parameter 1^n. The challenger runs the algorithm Setup, and obtains parameters *param*. Next, it obtains $(vk, sk) \leftarrow \mathsf{KeyGen}(param)$. C gives 1^n, *param*, and vk to the adversary.

Learning Phase: The adversary queries to the oracle SIGN.

- The oracle SIGN receives a message msg_i in the i-th query. It returns $\sigma_i \leftarrow \mathsf{Sign}(param, sk, msg_i)$ to the adversary.

Challenge Phase: The adversary $\mathcal{A}$ outputs a message msg^* and a forged signature σ^*.

- If goal = euf then, the challenger checks that $msg^* \neq msg_i$ for any i and $\mathsf{Ver}(param, vk, msg^*, \sigma^*) = 1$, then C outputs 1. Otherwise, it outputs 0.

- If goal = seuf then, the challenger checks that $(msg^*, \sigma^*) \neq (msg_i, \sigma_i)$ for any i and $\mathsf{Ver}(param, vk, msg^*, \sigma^*) = 1$, then C outputs 1. Otherwise, it outputs 0.

Definition 11.1.3 (EUF-CMA and sEUF-CMA security)**.** Let SIG be a signature scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{goal-cma}}_{\mathsf{SIG},\mathcal{A}}(n) = \Pr\left[\mathbf{Exp}^{\text{goal-cma}}_{\mathsf{SIG},\mathcal{A}}(n) = 1\right].$$

We say that SIG is existential unforgeable under chosen message attacks if $\mathbf{Adv}^{\text{euf-cma}}_{\mathsf{SIG},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

We say that SIG is strongly existential unforgeable under chosen message attacks if $\mathbf{Adv}^{\text{seuf-cma}}_{\mathsf{SIG},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

11.2 General Conversions to Secure Signature Schemes

We have found a lack of textbooks or notes containing the general conversion techniques to obtain secure signature schemes. We here give a survey of them. We hope that the textbook on this issue to appear.

11.2.1 From One-Way Function Family to Strong One-Time Signature Schemes

It is well-known that one-way functions yield one-time signature schemes and the obtained signature scheme is employed anywhere of cryptography. The first one is due to Lamport [Lam79] and we introduce only it in this section.

Suppose that $\mathsf{OWF} = (\mathsf{OWF.Setup}, \mathsf{OWF.Eval})$ is a one-way function family. $\mathsf{OWF.Setup}(1^n)$ outputs an index $a \in K_n$ that defines a function $f_a : D_{n,a} \to R_{n,a}$. For $x \in D_{n,a}$, $\mathsf{OWF.Eval}(a, x)$ outputs $f_a(x)$.

Scheme 11.2.1 (Lamport-OTS)**.** The message space of the signature is $\{0, 1\}^n$.

$\mathsf{Setup}(1^n)$**:** Given the security parameter 1^n, invoke $a \leftarrow \mathsf{OWF.Setup}(1^n)$ and outputs $param = a$.

$\mathsf{KeyGen}(a)$**:** Given the index a, it first choose $2n$ random elements $x_i^{(b)} \leftarrow D_{n,a}$ for $i \in [n]$ and $b \in \{0, 1\}$. It next computes $y_i^{(b)} \leftarrow f_a(x_i^{(b)})$. The signing key is $sk = X = \{x_i^{(b)}\}_{i,b}$. The verification key is $vk = Y = \{y_i^{(b)}\}_{i,b}$.

$\mathsf{Sign}(a, X, msg)$**:** The message is an n-bit string. Let msg_i denote the i-th bit of msg. Then, the algorithm reveals $x_i^{(msg_i)}$ as σ. Formally, it outputs $\sigma = \{x_i^{(msg_i)}\}_{i\in[n]}$.

$\mathsf{Ver}(a, Y, msg, \sigma)$**:** Parse $\sigma = \{\sigma_i\}_{i\in[n]}$. It checks that $y_i^{(msg_i)} = f_a(\sigma_i)$ and $\sigma_i \in D_{n,a}$. It accepts if all the checks are passed and rejects otherwise.

Theorem 11.2.2. *The scheme* Lamport-OTS *is one-time secure if* OWF *is one-way.*

For the proof, see, e.g., [BDS08, Section 7.2].

Direct use of one-way function can achieve only one-time security rather than strong one-time security. It is known that a target collision-resistant hash family (as known as a universal one-way hash family)[1] suffices to construct strongly one-time secure signature schemes [NY89]. Rompel showed that a one-way function family is suffice to construct a target collision-resistant hash family [Rom90] (see the concrete proof Katz and Koo [KK05]). In addition, if we replace the internal function of Lamport-OTS with a collision-resistant hash function, the similar proof shows the security of the obtained scheme. See the next section.

From Collision-Resistant Hash Family to Strong One-Time Signature Schemes

Replacing the one-way function family with the collision-resistant hash family, we can prove strong one-time security of the scheme Lamport-OTS.

Suppose that Hash $=$ (Hash.Setup, Hash.Eval) is a collision-resistant hash family with domain D_n and range R_n. Hash.Setup(1^n) outputs an index $a \in K_n$ that defines a function $h_a : D_n \to R_n$. For $x \in D_n$, Hash.Eval(a, x) outputs $h_a(x)$.

Scheme 11.2.3 (Lamport-OTS$'$)**.** The message space of the signature is $\{0, 1\}^n$.

Setup(1^n)**:** Given the security parameter 1^n, invoke $a \leftarrow$ OWF.Setup(1^n) and outputs $param = a$.

KeyGen(a)**:** Given the index a, it first choose $2n$ random elements $x_i^{(b)} \leftarrow D_{n,a}$ for $i \in [n]$ and $b \in \{0, 1\}$. It next computes $y_i^{(b)} \leftarrow f_a(x_i^{(b)})$. The signing key is $sk = X = \{x_i^{(b)}\}_{i,b}$. The verification key is $vk = Y = \{y_i^{(b)}\}_{i,b}$.

Sign(a, X, msg)**:** The message is an n-bit string. Let msg_i denote the i-th bit of msg. Then, the algorithm reveals $x_i^{(msg_i)}$ as σ. Formally, it outputs $\sigma = \{x_i^{(msg_i)}\}_{i\in[n]}$.

Ver(a, Y, msg, σ)**:** Parse $\sigma = \{\sigma_i\}_{i\in[n]}$. It checks that $y_i^{(msg_i)} = f_a(\sigma_i)$ and $\sigma_i \in D_{n,a}$. It accepts if all the checks are passed and rejects otherwise.

Theorem 11.2.4. *Suppose that $|D_n| / |R_n| = 2^{\omega(\log n)}$. The scheme* Lamport-OTS$'$ *is strongly one-time secure if* Hash *is collision resistant.*

Proof. Suppose that there exists an adversary $\mathcal{A}$ that wins the strong one-time security game. We construct an adversary $\mathcal{B}$ that outputs a collision.

At the first, $\mathcal{B}$ is given an index $a \leftarrow$ Hash.Setup(1^n). Then, it makes the signing key $X = \{x_i^{(b)}\}$ and the verification key $Y = \{y_i^{(b)}\}$ for $i \in [n]$ and $b \in \{0, 1\}$, where $y_i^{(b)} \leftarrow h_a(x_i^{(b)})$. $\mathcal{B}$ feeds a and Y to $\mathcal{A}$. If $\mathcal{A}$ queries a message msg to be

[1] Consider the following game; (1) the adversary $\mathcal{A}$ first output $msg \in D_n$, (2) the challenger C generates $a \leftarrow K_n$ and feeds it $\mathcal{A}$, (3) $\mathcal{A}$ outputs $msg' \in D_n$, (4) C outputs 1 if $msg \neq msg'$ and $h_a(msg) = h_a(msg')$ and 0 otherwise. The adversary wins if C outputs 1. We say $\mathcal{H} = \{\mathcal{H}_n\}_n$ (or corresponding Hash) is target collision resistance if no polynomial-time adversary wins the game with non-negligible probability.

signed, then $\mathcal{B}$ signs it by using the signing key X and return $\sigma = \{\sigma_i\} = \{x_i^{(msg_i)}\}$ to $\mathcal{A}$. At the end, $\mathcal{A}$ outputs (msg^*, σ^*).

We should consider two cases, (1) $msg^* = msg$ and $\sigma^* \neq \sigma$, and (2) $msg^* \neq msg$. We show $\mathcal{B}$ can output a collision with high probability in each case if $\mathcal{A}$ wins the game.
(1) $msg^* = msg$ and $\sigma^* \neq \sigma$: There must be an index i such that $\sigma_i^* \neq \sigma_i = x_i^{(msg_i)}$. If $\mathsf{Ver}(a, Y, msg^*, \sigma^*) = 1$, $h_a(\sigma_i^*) = h_a(\sigma_i)$. Then, $\mathcal{B}$ can output a collision (σ_i^*, σ_i) for h_a.
(2) $msg^* \neq msg$: There is an index i such that $msg_i^* \neq msg_i$. Notice that $\mathcal{A}$ have not seen $x_i^{(msg_i^*)}$. Hence, with high probability, $\sigma_i^* \neq x_i^{(msg_i^*)}$ because $\mathcal{B}$ chooses $x_i^{(msg_i^*)}$ uniformly at random from D_n and $|D_n| \,/\, |R_n| = 2^{\omega(\log n)}$. Therefore, $\mathcal{B}$ can output a collision $(\sigma_i^*, x_i^{(msg_i^*)})$ for h_a. □

We note that we can change Hash with a collision-resistant preimage sampleable functions PSF. The proof is essentially same.

11.2.2 From One-time Signature Scheme

Merkle's tree can be used to construct of EUF-CMA secure signature from one-time signature. Intuitively, the verification key of the scheme authenticates the verification keys of one-time signature. For theoretical and implementation techniques, see the survey by Buchmann, Dahmen, and Szydlo [BDS08].

11.2.3 From One-Way Trapdoor Permutations

We say the scheme TDOWF is a one-way trapdoor permutation scheme if each function f_a is a permutation (that is, the function f_a is one-to-one and the domain $D_{n,a}$ and the range $R_{n,a}$ are the same set) and write it TDOWP.

The security of the full-domain hash (FDH) paradigm (or well-known heuristic) is proved by Bellare and Rogaway [BR93] with introducing the random oracle paradigm (see also [BR96] and Coron [Cor00]). The probabilistic full-domain hash (PFDH) paradigm is a variant of FDH, which employs "salts", and proposed by Coron [Cor02, Appendix A].

Scheme 11.2.5 (TDOWP-FDH)**.** Let $\mathsf{TDOWP} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{Inv})$ be a one-way trapdoor permutation scheme with domain and range $D_{n,a} = R_{n,a}$. The hash function $H : \{0,1\}^* \to R_{n,a}$ is modeled as the random oracle. The signature scheme TDOWP-FDH is defined as follows:

$\mathsf{Setup}(1^n)$**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter from the arguments of the algorithms for ease of notation.

$\mathsf{KeyGen}(1^n)$**:** The key-generation algorithm invokes $\mathsf{TDOWP.TrapGen}(1^n)$ and obtains (a, t). It outputs $(vk = a, sk = t)$.

$\mathsf{Sig}(t, msg)$**:** The signing algorithm, given msg, first obtains $y = H(msg) \in R_{n,a}$. Then, it invokes $x \leftarrow \mathsf{TDOWP.Inv}(t, y)$ and outputs $x \in D_{n,a}$.

$\mathsf{Ver}(a, msg, \sigma = x)$**:** The verification algorithm first computes $y = H(msg)$. If $y = \mathsf{TDOWP.Eval}(a, x)$ it outputs 1, otherwise outputs 0.

Scheme 11.2.6 (TDOWP-PFDH)**.** Let $\mathsf{TDOWP} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{Inv})$ be a one-way trapdoor permutation scheme with domain and range $D_{n,a} = R_{n,a}$. The hash function $H : \{0,1\}^* \to R_{n,a}$ is modeled as the random oracle. The signature scheme TDOWP-PFDH is defined as follows:

$\mathsf{Setup}(1^n)$**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter from the arguments of the algorithms for ease of notation.

$\mathsf{KeyGen}(1^n)$**:** The key-generation algorithm invokes $\mathsf{TDOWP.TrapGen}(1^n)$ and obtains (a, t). It outputs $(vk = a, sk = t)$.

$\mathsf{Sig}(t, msg)$**:** The signing algorithm, given msg, first generates $r \leftarrow \{0,1\}^n$ and obtains $y = H(msg \circ r) \in R_{n,a}$. Then, it invokes $\mathsf{TDOWP.Inv}(t, y)$ and obtains $x \in D_n$. The signature is (r, x).

$\mathsf{Ver}(a, msg, \sigma = (r, x))$**:** The verification algorithm first computes $y = H(msg \circ r)$. If $y = \mathsf{TDOWP.Eval}(a, x)$ it outputs 1, otherwise outputs 0.

Theorem 11.2.7. *Both* TDOWP-FDH *and* TDOWP-PFDH *are EUF-CMA secure in the random oracle model if* TDOWP *is one-way.*

Roughly speaking, the adversary against TDOWP programs y as $H(msg_i)$ for some msg_i, which is queried to the random oracle. In order to obtain tighter security reductions, several researchers proposed the simulation and programming methods of the random oracle. The simplest one is the simulator chooses $i \in [Q]$ and setting y as $H(msg_i)$, where Q is the number of the queries to the random oracle H by the adversary. Using this method, we can upper bound

$$\mathbf{Adv}^{\text{euf-cma}}_{\mathsf{TDOWP\text{-}FDH},\mathcal{A}}(n) \leq \frac{1}{Q}\mathbf{Adv}_{\mathsf{TDOWP}}(n) + \mathsf{negl}(n),$$

where $\mathbf{Adv}_{\mathsf{TDOWP}}(n)$ is the upper bound of the advantages $\mathbf{Adv}^{\text{ow}}_{\mathsf{TDOWP},\mathcal{A}}(n)$ for any polynomial-time adversary $\mathcal{A}$ against one-way property of TDOWP. See, e.g., Coron [Cor00] to reduce the factor $1/Q$. We note that Coron's method can be applied to homomorphic functions.

11.2.4 From Collision-Resistant Preimage Sampleable Functions

Gentry et al. [GPV08] observed that we can replace OWTDP with a collision-resistant preimage sampleable function family PSF. Furthermore, it yields tighter security than that for OWTDP-FDH and OWTDP-PFDH. The schemes PSF-FDH and PSF-PFDH are defined as follows:

Scheme 11.2.8 (PSF-FDH)**.** Let $\mathsf{PSF} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SampleDom}, \mathsf{SamplePre})$ be a preimage sampleable function scheme with domain D_n and range R_n. The hash function $H : \{0,1\}^* \to R_n$ is modeled as the random oracle. The scheme PSF-FDH is defined as follows:

$\mathsf{Setup}(1^n)$**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter for ease of notation.

$\mathsf{KeyGen}(1^n)$**:** The key-generation algorithm invokes $\mathsf{PSF.TrapGen}(1^n)$ and obtains (a, t). It outputs $(vk = a, sk = t)$.

$\mathsf{Sig}(sk = t, msg)$**:** The signing algorithm, given msg, first checks its local storage. If (msg, σ_{msg}) is in local storage, the output σ_{msg}. Else, it computes $y \leftarrow H(msg) \in R_n$, invokes $\mathsf{PSF.SamplePre}(t, y)$ and obtains $\sigma_{msg} = x \in D_n$.

$\mathsf{Ver}(vk = a, msg, \sigma = x)$**:** The verification algorithm first computes $y = H(msg)$. If $y = \mathsf{PSF.Eval}(a, x)$ it outputs 1, otherwise outputs 0.

Scheme 11.2.9 (PSF-PFDH)**.** Let $\mathsf{PSF} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SampleDom}, \mathsf{SamplePre})$ be a preimage sampleable function scheme with domain D_n and range R_n. The hash function $H : \{0,1\}^* \to R_n$ is modeled as the random oracle. The scheme PSF-PFDH is defined as follows:

$\mathsf{Setup}(1^n)$**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter for ease of notation.

$\mathsf{KeyGen}(1^n)$**:** The key-generation algorithm invokes $\mathsf{PSF.TrapGen}(1^n)$ and obtains (a, t). It outputs $(vk = a, sk = t)$.

$\mathsf{Sig}(sk = t, msg)$**:** The signing algorithm, given msg, first generates $r \leftarrow \{0,1\}^n$ and obtains $y = H(msg \circ r) \in R_n$. Then, it invokes $\mathsf{PSF.SamplePre}(t, y)$ and obtains $x \in D_n$. The signature is (r, x).

$\mathsf{Ver}(vk = a, msg, \sigma = (r, x))$**:** The verification algorithm first computes $y = H(msg \circ r)$. If $y = \mathsf{PSF.Eval}(a, x)$ it outputs 1, otherwise outputs 0.

Since the underlying functions are collision resistant, the security reduction is tighter than that in the FDH signature based on one-way trapdoor functions. We note that this idea is used independently by Bernstein [Ber08] to given the tighter security reduction for the Rabin–Williams signature scheme, which is based on the 4-to-1 one-way function.

Theorem 11.2.10 (Propositions 6.1 and 6.2, [GPV08])**.** *Let* PSF *be a collision-resistant preimage sampleable functions with domain D_n and range R_n. Then both* PSF-FDH *and* PSF-PFDH *are sEUF-CMA secure in the random oracle model.*

11.2.5 From Identification Schemes: The Fiat–Shamir Conversion

Let us recall canonical identification schemes. The scheme has a prover $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ and a verifier $\mathsf{V} = (\mathsf{V}_1, \mathsf{V}_2)$. The interaction between them is a triplet of a commitment y from P_1, a challenge c from V_1, and a response z from P_2. Roughly speaking, if we replace c by V_1 with $c = H(msg, y)$, there is no interaction. The signer computes $y \leftarrow \mathsf{P}_1$, $c \leftarrow H(msg, y)$, and $z \leftarrow \mathsf{P}_2$ and sets (y, c, z) as a signature. Then, verifier can verify it by invoking V_2.

Originally, Fiat and Shamir introduced the conversion as a heuristic. Its security is proved in Pointcheval and Stern [PS96] and Okamoto and Ohta [OO98].

Finally, Abdalla, An, Bellare, and Namprempre minimized the assumption on the underlying ID scheme, which must be passively secure and has large challenge space.

Recall that a canonical identification in Section 6.2.2. We describe the slightly modified version of the Fiat–Shamir conversion in [AABN08].

Scheme 11.2.11 (The Fiat–Shamir conversion with a slight modification [AABN08])**.** Let $\mathsf{ID} = (\mathsf{ID.Setup}, \mathsf{ID.KeyGen}, \mathsf{ID.P} = (\mathsf{ID.P}_1, \mathsf{ID.P}_2), \mathsf{ID.V} = (\mathsf{ID.V}_1, \mathsf{ID.V}_2))$ be a canonical ID scheme with challenge space C_n such that $|C_n| = 2^{\omega(\log n)}$. Let $H : \{0,1\}^* \to C_n$ be the random oracle. Let $\alpha(n)$ be some non-negative integer function. The converted signature scheme $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ is defined as follows:

$\mathsf{Setup}(1^n)$**:** Given the security parameter, it obtains $param \leftarrow \mathsf{ID.Setup}(1^n)$ and outputs $param' = param$

$\mathsf{KeyGen}(param')$**:** Given the public parameter $param$, it obtains $(pk, sk) \leftarrow \mathsf{ID.KeyGen}(1^n)$ and outputs $(vk', sk') = (pk, (pk, sk))$.

$\mathsf{Sign}(param', sk' = (pk, sk), msg)$**:** Given $param'$, pk, sk, and a message msg, it generate a signature as follows: First, it obtains $(cmt, st_{\mathsf{P}}) \leftarrow \mathsf{ID.P}_1(param, pk, sk)$. Next, it chooses $r \leftarrow \{0,1\}^{\alpha(n)}$ and computes $ch \leftarrow H(r \circ cmt \circ msg)$. Finally, it obtains $rsp \leftarrow \mathsf{ID.P}_2(ch, st_{\mathsf{P}})$. It outputs $\sigma = (r, cmt, rsp)$.

$\mathsf{Ver}'(param', vk' = pk, msg, \sigma = (r, cmt, rsp))$**:** Given $param'$, vk', msg, and σ, it makes a decision as follows: It computes $ch \leftarrow H(r \circ cmt \circ msg)$, obtains $dec \leftarrow \mathsf{ID.V}_2(pk, cmt, ch, rsp)$, and outputs dec.

If $s(n) = 0$ for any n, this is the original Fiat–Shamir conversion. To make the result general, Abdalla et al. introduced $\alpha(n)$ as the parameter.

To describe the security, we require the non-triviality of an ID scheme. That is, commitments generated by P_1 has sufficiently large min-entropy $\beta(n) = \omega(\log n)$.

Theorem 11.2.12 ([AABN08])**.** *Suppose that* ID *is a passively-secure canonical identification scheme with* $|C_n| = 2^{\omega(\log n)}$ *and* $\alpha(n) + \beta(n) = \omega(\log(n))$*. Then, the obtained signature scheme* Sig' *is EUF-CMA secure in the random oracle model.*

Corollary 11.2.13 ([AABN08])**.** *Suppose that* ID *is a passively-secure, non-trivial, canonical identification scheme with* $|C_n| = 2^{\omega(\log n)}$*. Suppose that* $s(n) = 0$ *for any* n*. Then, the obtained signature scheme* Sig' *is EUF-CMA secure in the random oracle model.*

The Fiat–Shamir Conversion with Aborts

In the above conversion, ID has perfect completeness, that is, the decision of the legitimate verifier interacting the legitimate prover is 1 with probability 1. Interestingly, Lyubashevsky [Lyu09] observed that the ID schemes are not needed to be perfectly correct in the Fiat-Shamir conversion. The signer repeat the above procedure until it succeeds.

One-Time Signature Schemes from Identification Schemes: The Fiat–Shamir Conversion without the Random Oracle

In [BS08], Bellare and Shoup proved that, if we want to construct *one-time* signature[2], a collision-resistant hash family can be used instead of the random oracle in the Fiat–Shamir conversion. However, the underlying identification must be concurrently secure rather than passively secure as in the Fiat–Shamir conversion. In addition, the underlying identification must have a special soundness.

Intuitively, the public and secret key of the identification scheme corresponds to the master verification and signing key, respectively. The commitment generated by the prover and its randomness are the session verification key and the session signing key. For each session key, only a message can be signed. If only one session is allowed, the scheme is a one-time signature scheme.

Scheme 11.2.14 (The Fiat–Shamir conversion without the random oracle [BS08])**.** Let $\mathsf{ID} = (\mathsf{ID.Setup}, \mathsf{ID.KeyGen}, \mathsf{ID.P} = (\mathsf{ID.P}_1, \mathsf{ID.P}_2), \mathsf{ID.V} = (\mathsf{ID.V}_1, \mathsf{ID.V}_2))$ be a canonical ID scheme with challenge space C_n such that $|C_n| = 2^{\omega(\log n)}$. Let $\mathsf{Hash} = (\mathsf{Hash.Setup}, \mathsf{Hash.Eval})$ be a hash scheme. Let $\alpha(n)$ be some non-negative integer function. The converted signature scheme $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ is defined as follows:

$\mathsf{Setup}(1^n)$**:** Given the security parameter, it obtains $param \leftarrow \mathsf{ID.Setup}(1^n)$ and $k \leftarrow \mathsf{Hash.Setup}(1^n)$, and outputs $param' = (param, k)$.

$\mathsf{KeyGen}(param' = (param, k))$**:** Given the public parameter $param'$, it obtains $(pk, sk) \leftarrow \mathsf{ID.KeyGen}(1^n)$. Next, it obtains $(cmt, st_{\mathsf{P}}) \leftarrow \mathsf{ID.P}_1(param, pk, sk)$. It outputs $(vk', sk') = ((pk, cmt), (pk, sk, st_{\mathsf{P}}))$.

$\mathsf{Sign}(param', sk' = (pk, sk, st_{\mathsf{P}}), msg)$**:** Given $param', pk, sk$, and a message msg, it generate a signature as follows: It computes $ch \leftarrow \mathsf{Hash.Eval}(k, cmt \circ msg) = h_k(cmt \circ msg)$. Finally, it obtains $rsp \leftarrow \mathsf{ID.P}_2(ch, st_{\mathsf{P}})$. It outputs $\sigma = rsp$.

$\mathsf{Ver}(param', vk' = (pk, cmt), msg, \sigma = rsp)$**:** Given $param'$, vk', msg, and σ, it makes a decision as follows: It computes $ch \leftarrow \mathsf{Hash.Eval}(k, cmt \circ msg)$, obtains $dec \leftarrow \mathsf{ID.V}_2(pk, cmt, ch, rsp)$, and outputs dec.

Theorem 11.2.15 ([BS08])**.** *Suppose that* ID *is a concurrently-secure, canonical, and special-sound identification scheme with* $|C_n| = 2^{\omega(\log n)}$*. Also, suppose that* Hash *is collision-resistant. Then, the obtained signature scheme* Sig' *is strongly one-time secure.*

[2] Precisely, they defined *two-tier* signature and showed it includes one-time signature as a special version.

11.2.6 From Trapdoor Hash Schemes and Weakly Secure Signature Schemes

Shamir and Tauman showed that the hash-sign-switch paradigm bares secure signature scheme.

Scheme 11.2.16 (The Hash-Sign-Switch Paradigm [ST01]). Let $\mathsf{THash} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SampleDom}, \mathsf{TrapCol})$ be a trapdoor hash scheme and let $\mathsf{Sig} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ be a weakly-secure signature scheme. A new signature scheme $\mathsf{Sig}' = (\mathsf{Setup}', \mathsf{KeyGen}', \mathsf{Sign}', \mathsf{Ver}')$ is obtained as follows:

$\mathsf{Setup}'(1^n)$**:** Given the security parameter 1^n, invoke $param \leftarrow \mathsf{Setup}(1^n)$ and output $param$.

$\mathsf{KeyGen}'(param)$**:** Generate key pairs $(vk, sk) \leftarrow \mathsf{KeyGen}(param)$ and $(a, t) \leftarrow \mathsf{TrapGen}(1^n)$. The new key pair is $vk' = (vk, a)$ and $sk' = (sk, a, t)$.

$\mathsf{Sign}' = (\mathsf{SignOff}, \mathsf{SignOn})$**:** The signing algorithm has two stages, off-line phase and one-line phase.

$\mathsf{SignOff}'(sk')$**:** Choose a random message $msg' \leftarrow M_n$ and a random string $r' \leftarrow \mathsf{SampleDom}(1^n)$. Then, compute a digest $d = h_a(msg', r')$ by $\mathsf{Eval}(a, msg', r')$. Next, obtain $\sigma_{\mathrm{off}} \leftarrow \mathsf{Sign}(sk, d)$. It outputs σ_{off} and $st_{\mathrm{off}} = (d, msg', r')$.

$\mathsf{SignOn}'(sk', \sigma_{\mathrm{off}}, st_{\mathrm{off}}, msg)$**:** Obtain $r \leftarrow \mathsf{TrapCol}(a, t, msg', r', msg_1)$. Output $\sigma = (\sigma_{\mathrm{off}}, r)$ as the signature.

$\mathsf{Ver}'(vk' = (vk, a), msg, \sigma = (\sigma_{\mathrm{off}}, r))$**:** Compute $d \leftarrow h_a(msg, r)$ and output $dec \leftarrow \mathsf{Ver}(vk, d, \sigma_{\mathrm{off}})$ as the decision.

Theorem 11.2.17. *Let* THash *be a trapdoor hash scheme and let* Sig *be an EUF-wCMA secure signature scheme. Then,* Sig' *is an EUF-CMA secure signature scheme.*

Although we have changed the definition of trapdoor hash families, we can give the security proof by the essentially similar way to the one of Shamir and Tauman. Hence, we omit the details.

11.2.7 From Identity-based Encryption

As noted in the paper by Boneh and Franklin [BF03], Naor pointed out identity-based encryption induces the signature scheme. For notation and notions on identity-based encryption, see Chapter 14.

Given an IBE scheme $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$, a signature scheme $\mathsf{SIG} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Ver})$ is defined as follows;

$\mathsf{KeyGen}(1^n)$**:** $(vk, sk) = (param, msk) \leftarrow \mathsf{Setup}(1^n)$.

$\mathsf{Sig}(msg, sk)$**:** $\sigma = usk_{msg} \leftarrow \mathsf{Ext}(msg, msk)$.

$\mathsf{Ver}(msg, \sigma, vk)$**:** To verify the signature σ on msg under the verification key vk, it randomly generates a ciphertext $ct \leftarrow \mathsf{Enc}(m, msg, pk)$ with a random plaintext m, and checks that $m = \mathsf{Dec}(ct, \sigma, pk)$. If the check is passed, it accepts the signature. Otherwise, reject.

It is easy to show that this construction yields secure signature scheme if the underlying identity-based encryption scheme is fully-ID secure. Roughly speaking, if there is a forger for the signature scheme, one can extract the identity and its decryption key. Hence, this violates the security of the underlying identity-based encryption scheme.

11.3 The Gentry–Peikert–Vaikuntanathan Signature

We now turn back to lattice-based signature schemes. The first appearing ones are proposed by Gentry et al. [GPV08]. The schemes are obtained by applying FDH and PFDH paradigms to LPSF.

11.3.1 Description

Applying PSF-FDH and PSF-PFDH to the lattice-based CR-PSFs in Section 10.5, we obtained a sEUF-CMA secure signature scheme based on the SIS assumption.

Scheme 11.3.1 (GPV-FDH)**.** We model the hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ as the random oracle.

$\mathsf{Setup}(1^n)$**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter for ease of notation.

$\mathsf{KeyGen}(1^n)$**:** The key-generation algorithm invokes $\mathsf{LPSF.TrapGen}(1^n)$ in Section 10.5 and obtains $(\boldsymbol{A}, \boldsymbol{T}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}^{m\times m}$. It outputs $(vk = \boldsymbol{A}, sk = \boldsymbol{T})$.

$\mathsf{Sign}(sk = \boldsymbol{T}, msg)$**:** The signing algorithm, given msg, first obtains $\boldsymbol{u} = H(msg)$. Then, it invokes $\mathsf{LPSF.SamplePre}(\boldsymbol{A}, \boldsymbol{T}, s, \boldsymbol{u})$ in Section 10.5 and obtains $\boldsymbol{e}$. (Here, $\boldsymbol{e}$ is a sample from the distribution D which is statistically close to $D_{\Lambda_q^\perp(\boldsymbol{A}),s,\boldsymbol{t}}$ where $\boldsymbol{t}$ is any solution of $\boldsymbol{A}\boldsymbol{t} = \boldsymbol{u} \bmod q$.)

$\mathsf{Ver}(vk = \boldsymbol{A}, msg, \sigma = \boldsymbol{e})$**:** The verification algorithm first computes $\boldsymbol{u} = H(msg)$. Then, it verifies $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q$. Output 1 if the check is passed, otherwise, 0.

Scheme 11.3.2 (GPV-PFDH)**.** We model the hash function $H : \{0,1\}^* \to \mathbb{Z}_q^n$ as the random oracle.

$\mathsf{Setup}(1^n)$**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter for ease of notation.

$\mathsf{KeyGen}(1^n)$**:** The key-generation algorithm invokes $\mathsf{LPSF.TrapGen}(1^n)$ in Section 10.5 and obtains $(\boldsymbol{A}, \boldsymbol{T}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}^{m\times m}$. It outputs $(vk = \boldsymbol{A}, sk = \boldsymbol{T})$.

$\mathsf{Sig}(sk = \boldsymbol{T}, msg)$**:** The signing algorithm, given msg, first generates a random string $r \leftarrow \{0,1\}^n$ and obtains $\boldsymbol{u} = H(msg \circ r)$. Then, it invokes

LPSF.SamplePre$(\boldsymbol{A}, \boldsymbol{T}, s, \boldsymbol{u})$ in Section 10.5 and obtains $\boldsymbol{e}$. It outputs $(r, \boldsymbol{e})$ as the signature.

Ver$(vk = \boldsymbol{A}, msg, \sigma = (r, \boldsymbol{e}))$**:** The verification algorithm first computes $\boldsymbol{u} = H(msg \circ r)$. Then, it verifies $\boldsymbol{Ae} = \boldsymbol{u} \bmod q$. Output 1 if the check is passed, otherwise, 0.

Theorem 11.3.3 (Security, [GPV08])**.** *Let $m \geq (5 + 3\delta)n \log q$ for some constant $\delta > 0$. Let $s \geq L \cdot \omega(\sqrt{\log m})$, where $L = O(\sqrt{n \log q})$. The above schemes are sEUF-CMA secure if $\mathrm{SIS}_{q,m,2s\sqrt{m}}$ is hard.*

The security proofs of two schemes is obtained by combining the arguments in Section 10.5 and Section 11.2.4.

11.4 The Stehlé–Steinfeld–Tanaka–Xagawa Signature

As an analogue of the GPV signatures, we introduce the SSTX signatures based on the ideal-lattice-based PSFs. These schemes are obtained by replacing the underlying PSFs LPSF with ILPSF.

11.4.1 Description

Scheme 11.4.1 (SSTX-PFDH)**.** We model the hash function $H : \{0,1\}^* \to R_{\mathbf{f},q}$ as the random oracle.

Setup(1^n)**:** The setup algorithm outputs $param = 1^n$. We omit the public parameter for ease of notation.

KeyGen(1^n)**:** The key-generation algorithm invokes ILPSF.TrapGen(1^n) in Section 10.5 (hence, ExtIdLattice in Section 10.3) and obtains $(\check{\boldsymbol{a}}, \boldsymbol{T})$. It outputs $(vk = \check{\boldsymbol{a}}, sk = \boldsymbol{T})$.

Sig$(sk = \boldsymbol{T}, msg)$**:** The signing algorithm, given msg, first generates a random string $r \leftarrow \{0,1\}^n$ and obtains $\mathbf{u} = H(msg \circ r)$. Then, it invokes ILPSF.SamplePre$(\check{\boldsymbol{a}}, \boldsymbol{T}, s, \mathbf{u})$ in Section 10.5 and obtains $\check{\boldsymbol{e}}$. It outputs $(r, \check{\boldsymbol{e}})$ as the signature.

Ver$(vk = \check{\boldsymbol{a}}, msg, \sigma = (r, \check{\boldsymbol{e}}))$**:** The verification algorithm first computes $\mathbf{u} = H(msg \circ r)$. Then, it verifies . If $\check{\boldsymbol{a}}\check{\boldsymbol{e}} = \mathbf{u} \bmod q$ it outputs 1, otherwise outputs 0.

11.4.2 Security Proofs

The security proofs of two schemes is obtained by combining the arguments in Section 10.5 and Section 11.2.4.

11.5 The Lyubashevsky–Micciancio One-Time Signature

This scheme has a flavor of the Bellare–Shoup transformation with the Lyubashevsky identification scheme, $\mathsf{Ly08\text{-}ID_{C/IL}}$, but there are two main differences. The first difference is the change of key spaces and the method of key generation to obtain perfect correctness. The second difference is no use of collision resistant hash functions. We note that this is a reverse-chronicle order. Lyubashevsky and Micciancio [LM08] proposed the scheme in March 2008 and Lyubashevsky [Lyu08b, Lyu09] proposed his identification scheme in September 2008 and December 2009.

11.5.1 Description

For simplicity, we fix $\mathbf{f} = x^n + 1$ and $l = \lfloor \log^2 n \rfloor$. Let us fix $q = n^3$ and $m \leftarrow \lceil \log n \rceil$. Define for any $i \in \mathbb{N}$,

$$D_{e,i} = \{\check{\boldsymbol{y}} \in R_{\mathbf{f},q}^m \mid \|\check{\boldsymbol{y}}\|_\infty \le 5iq^{1/m}\} \text{ and } D_{r,i} = \{\check{\boldsymbol{y}} \in R_{\mathbf{f},q}^m \mid \|\check{\boldsymbol{y}}\|_\infty \le 5inpq^{1/m}\}.$$

We also define

$$G = \{\check{\boldsymbol{y}} \in R_{\mathbf{f},q}^m \mid \|\check{\boldsymbol{y}}\|_\infty \le 10q^{1/m} n \log^2 n\}.$$

Scheme 11.5.1 (LM-OTS [LM08])**.**

$\mathsf{Setup}(1^n)$**:** It chooses a random row vector $\check{\boldsymbol{a}} \leftarrow R_{\mathbf{f},q}^m$ uniformly and outputs it.

$\mathsf{KeyGen}(\check{\boldsymbol{a}})$**:** It first choose $r \in \{0,1\}^l$. If $r = 0^l$ set $j = l$. Else, set j as the position of the left-most standing bit of r. Pick $\check{\boldsymbol{e}} \leftarrow D_{e,j}$ and $\check{\boldsymbol{r}} \leftarrow D_{r,j}$. Compute $\mathbf{u} \leftarrow h_{\check{\boldsymbol{a}}}(\check{\boldsymbol{e}})$ and $\mathbf{y} \leftarrow h_{\check{\boldsymbol{a}}}(\check{\boldsymbol{r}})$. The signing key is $(\mathbf{u}, \mathbf{y}, \check{\boldsymbol{e}}, \check{\boldsymbol{r}})$ and the verification key is $(\mathbf{u}, \mathbf{y})$.

$\mathsf{Sign}(\check{\boldsymbol{a}}, sk = (\mathbf{u}, \mathbf{y}, \check{\boldsymbol{e}}, \check{\boldsymbol{r}}), msg = \mathbf{c})$**:** The message $\mathbf{c}$ is in $\{-1, 0, +1\}^n$. It outputs $\sigma = \check{\boldsymbol{z}} \leftarrow \mathbf{c} \otimes \check{\boldsymbol{e}} + \check{\boldsymbol{r}}$.

$\mathsf{Ver}(\check{\boldsymbol{a}}, vk = (\mathbf{u}, \mathbf{y}), msg = \mathbf{c}, \sigma = \check{\boldsymbol{z}})$**:** If $\check{\boldsymbol{z}} \in G$ and $h_{\check{\boldsymbol{a}}}(\check{\boldsymbol{z}}) = \mathbf{c} \otimes \mathbf{u} + \mathbf{y}$ then output 1 (accept) and output 0 otherwise.

Theorem 11.5.2 ([LM08, Theorem 8])**.** *The above scheme is strongly one-time secure if the* $\mathbf{f}\text{-SIS}_{q,m,\beta}^\infty$ *is hard on average, where* $\beta = 20q^{1/m} n \log^2 n$*. In particular, for appropriately settings on the parameters, the scheme is secure if* $\mathbf{f}\text{-SVP}_\gamma^\infty$ *is hard in the worst case, where* $\gamma = \tilde{O}(n^2)$.

For the proof, see the original paper [LM08].

11.6 The Lyubashevsky Signature

We also obtain an EUF-CMA secure signature by applying the Fiat-Shamir transformation with aborts to the basic protocol of $\mathsf{Ly09\text{-}ID}$ Section 6.8..

11.6.1 Descriptions

Scheme 11.6.1 (Ly-SIG, [Lyu09])**.** All the participant agree with the parameters m, q, D_e, D_r, D_c, and G as in Section 6.8. We model the hash function $H : \{0,1\}^* \to D_c$ as the random oracle. Let $\mathbf{f}$ denote $x^n + 1$.

Setup(1^n)**:** It outputs $\check{\boldsymbol{a}} \leftarrow R_{\mathbf{f},q}^m$ uniformly at random.

KeyGen$(\check{a})$**:** It chooses $\check{\boldsymbol{e}} \leftarrow D_e$ and computes $\mathbf{u} = h_{\check{a}}(\check{\boldsymbol{e}})$. It outputs $sk = (\mathbf{u}, \check{\boldsymbol{e}})$ and $vk = \mathbf{u}$.

Sig$(\check{a}, (\mathbf{u}, \check{e}), msg)$**:** Choose $\check{\boldsymbol{r}} \leftarrow D_r$ and compute $\mathbf{y} \leftarrow h_{\check{a}}(\check{\boldsymbol{r}})$. Compute $\mathbf{c} \leftarrow H(msg \circ \mathbf{y})$. Then compute $\check{z} \leftarrow \mathbf{c} \otimes \check{\boldsymbol{e}} + \check{\boldsymbol{r}}$. If $\check{z} \in G$ then it outputs $\sigma = (\mathbf{y}, \check{z})$. Otherwise repeat the above procedures.

Ver$(\check{a}, \mathbf{u}, msg, \sigma = (\mathbf{y}, \check{z}))$**:** First, compute $\mathbf{c} \leftarrow H(msg \circ \mathbf{y})$. If $\check{z} \in G$ and $h_{\check{a}}(\check{z}) = \mathbf{c} \otimes \mathbf{u} + \mathbf{y}$ outputs 1, otherwise outputs 0.

Theorem 11.6.2 ([Lyu09, Theorem 3])**.** *The above scheme is EUF-CMA secure if* $\mathbf{f}$-$\mathrm{SIS}_{q,m,\beta}^{\infty}$ *is hard on average, where* $\beta = 2(mn-1)\sigma\kappa$*. In particular, let* σ *be a constant and* $\kappa(n) = \Theta(\log^2 n)$*. Then for appropriately settings on the parameters, the scheme is secure if* $\mathbf{f}$-$\mathrm{SVP}_{\gamma}^{\infty}$ *is hard in the worst case, where* $\gamma = \tilde{O}(n^2)$*.*

11.7 The Signature from "Bonsai"

Very recently, Peikert proposed a signature scheme made by "Bonsai." This signature scheme has the flavors of the Hohenberger–Waters signature scheme based on the RSA assumption and as the IBE-to-Sig conversion.

11.7.1 Description

Scheme 11.7.1 (Bonsai-wSIG)**.** We model the hash function. The parameters are defined as follows: Two integers $m_1, m_2 = O(n \log q)$, and a bound $L = O(\sqrt{n \log q})$. Let a hashed message length be k and total dimension $m = m_1 + (k+1)m_2$, and a Gaussian parameter $s = L \cdot \omega(\sqrt{\log n})$.

KeyGen(1^n)**:** Generate $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n \times (m_1+m_2)}$ with a basis $\boldsymbol{T}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_0)$ such that $\|\tilde{\boldsymbol{T}}\| \le L$. For each $(b, j) \in \{0,1\} \times [k]$, generate uniform and independent $\boldsymbol{A}_j^{(b)} \in \mathbb{Z}_q^{n \times m_2}$. Output $vk = (\boldsymbol{A}_0, \{\boldsymbol{A}_j^{(b)}\})$ and $sk = (\boldsymbol{S}, vk)$.

Sig$(sk = (\boldsymbol{T}, vk), msg = \boldsymbol{\mu} \in \{0,1\}^k)$**:** Let $\boldsymbol{A}_{\boldsymbol{\mu}} = [\boldsymbol{A}_0 | \boldsymbol{A}_1^{\mu_1} | \ldots | \boldsymbol{A}_k^{\mu_k}] \in \mathbb{Z}_q^{n \times m}$. Let $\boldsymbol{T}_{\boldsymbol{\mu}} \leftarrow$ ExtBasis$(\boldsymbol{T}, \boldsymbol{A}_\mu)$. Then, output $\sigma = \boldsymbol{e} \leftarrow$ SampleD$(\boldsymbol{T}_\mu, s, \mathbf{0})$.

Ver$(vk = (A_0, \{A_j^{(b)}\}), msg = \boldsymbol{\mu}, \sigma = \boldsymbol{e})$**:** Let $\boldsymbol{A}_{\boldsymbol{\mu}}$ as above. If $\boldsymbol{e} \neq \mathbf{0}$, $\|\boldsymbol{e}\| \le s \cdot \sqrt{m}$, and $\boldsymbol{A}_\mu \boldsymbol{e} = \mathbf{0} \bmod q$, output 1 as accept; otherwise, output 0 as reject.

Theorem 11.7.2 ([Pei09b])**.** *Let* $m = m_1 + (k+1)m_2$ *and* $m' = m_1 + (2k+1)m_2$*. The above scheme is EUF-wCMA secure if* $\mathrm{SIS}_{q,m',\beta}$ *is hard on average, where* $\beta = s\sqrt{m}$*.*

For the proof, see [Pei09b].

Combining the above scheme Bonsai-wSIG and the trapdoor commitment scheme, we obtain the scheme Bonsai-SIG. If we set $pk = a$ of the public key of the trapdoor-commitment scheme as the public parameter, we then have the signature scheme. In addition, if the verification key vk includes a and the signing key sk includes the corresponding trapdoor t, the scheme is now one-line/off-line signature.

Scheme 11.7.3 (Bonsai-SIG)**.** Let THash = (TrapGen, Eval, SampleDom, TrapCol) be a trapdoor hash scheme with range $\{0,1\}^k$. The parameters are defined as follows: Two integers $m_1, m_2 = O(n \log q)$, and a bound $L = O(\sqrt{n \log q})$. A hashed message length k and total dimension $m = m_1 + (k+1)m_2$, and a Gaussian parameter $s = L \cdot \omega(\sqrt{\log n})$.

KeyGen(1^n)**:** Invoke Bonsai-wSIG.KeyGen and obtain $vk' = (\boldsymbol{A}_0, \{\boldsymbol{A}_j^{(b)}\})$ and $sk' = (\boldsymbol{T}, vk')$. Invoke TrapGen($1^n$) and obtain $pk = a$ and $sk = t$. Output $vk = (\boldsymbol{A}_0, \{\boldsymbol{A}_j^{(b)}\}, a)$ and $sk = (\boldsymbol{T}, t, vk)$.

Sign = (SignOff, SignOn)**:**

SigOff($sk_{\text{off}} = (\boldsymbol{T}, vk)$)**:** First generate random string $r \leftarrow$ SampleDom(1^n) and compute $\boldsymbol{\mu} \leftarrow$ Eval($0; r$) $\in \{0,1\}^k$. Then, let $\boldsymbol{A}_{\boldsymbol{\mu}} = [\boldsymbol{A}_0 | \boldsymbol{A}_1^{\mu_1} | \ldots | \boldsymbol{A}_k^{\mu_k}] \in \mathbb{Z}_q^{n \times m}$ and let $\boldsymbol{T}_{\boldsymbol{\mu}} \leftarrow$ ExtBasis($\boldsymbol{T}, \boldsymbol{A}_{\boldsymbol{\mu}}$). Output $\sigma_{\text{off}} = \boldsymbol{e} \leftarrow$ SampleD($\boldsymbol{T}_{\boldsymbol{\mu}}, s, \boldsymbol{0}$) and $st_{\text{off}} = (\boldsymbol{\mu}, r)$.

SigOn($sk_{\text{on}} = t, \sigma_{\text{off}} = \boldsymbol{e}, st_{\text{off}} = (\boldsymbol{\mu}, r), msg \in \{0,1\}^*$)**:** It computes $r' \leftarrow$ TrapCol($\boldsymbol{\mu}, 0, r, msg$). Then outputs the signature $\sigma = (r', \boldsymbol{e})$.

Ver($vk = (\boldsymbol{A}_0, \{\boldsymbol{A}_j^{(b)}\}), msg, \sigma = (r, \boldsymbol{e})$)**:** First compute $\boldsymbol{\mu} \leftarrow$ Eval(msg, r). Let $\boldsymbol{A}_{\boldsymbol{\mu}}$ as above. If $\boldsymbol{e} \neq \boldsymbol{0}$, $\|\boldsymbol{e}\| \leq s \cdot \sqrt{m}$, and $\boldsymbol{A}_{\boldsymbol{\mu}} \boldsymbol{e} = \boldsymbol{0} \bmod q$, output 1 as accept; otherwise, output 0 as reject.

The signing algorithm can be split into two algorithms, SigOff and SigOn, where SigOff does not involve the message.

Remark 11.7.4. Notice that the above scheme is EUF-CMA secure but not sEUF-CMA secure; From a valid signature $\sigma = (r, \boldsymbol{e})$ on msg, a new signature $\sigma = (r, -\boldsymbol{e})$ on msg is obtained. To protect the scheme against the above attack, Rükert fixed the verification; the public key contains $\boldsymbol{u} \leftarrow \mathbb{Z}_q^n$, the signer samples $\boldsymbol{e} \leftarrow$ SampleD($\boldsymbol{T}_{\boldsymbol{\mu}}, s, \boldsymbol{u}$), and the verifier checks $\boldsymbol{A}_{\boldsymbol{\mu}} \boldsymbol{e} \equiv \boldsymbol{u} \bmod q$. For the details and the proofs, see Rükert's paper [Rüc10].

Remark 11.7.5. Stehlé et al. [SSTX09] already proposed ideal-lattice version of "Bonsai" (see Section 10.6). Replacing the "Bonsai" technique with the miniature "Bonsai," we obtain ideal-lattice-based "Bonsai" signature scheme.

12 Encryption

Organization: Section 12.1 reviews the brief history of lattice-based encryptions. Section 12.2 reviews the definitions of public-key encryption, that is, model and security notions. In Section 12.4 gave the review of the Ajtai–Dwork encryption scheme. Section 12.5 and Section 12.6 are the reviews of the Goldreich–Goldwasser–Halevi and NTRU encryption schemes. We review the Regev03 encryption scheme in Section 12.7. Regev's LWE-based encryption scheme, which plays important roles in lattice-based cryptography, is reviewed in Section 12.8. Section 12.9 reviews the "dual" of Regev's LWE-based encryption scheme. Section 12.10 contains lossy trapdoor functions and its children by Peikert and Waters. By using them, we describe the Peikert–Waters encryption scheme in Section 12.10.2.

12.1 Introduction

After the seminal result of Ajtai [Ajt96], Ajtai and Dwork [AD97] gave the first public-key encryption scheme AD based on the worst-case hardness of lattice problems. However, their inefficiency in the real world opened the door of attacks. Hence, efficient public-key encryption schemes with harder security reduction were needed.

The partial answers appeared almost simultaneously; GGH and NTRU. Goldreich, Goldwasser, and Halevi [GGH97b] proposed a lattice variant of the McEliece encryption scheme [McE78] (we describe them later). Hoffstein, Pipher, and Silverman also proposed a encryption scheme employing the quotient ring of polynomials [HPS98]. These schemes are more efficient than AD, but they lack the security proof based on the worst-case hardness of lattice problems.

	param	*ek*	Enc	C	$\lvert ek\rvert$	$\lvert ct\rvert/\lvert msg\rvert$
AD-GGH (12.4)	–	$\boldsymbol{A}, i_0, i_1$	$\boldsymbol{Ae} + \frac{\mu}{2}\boldsymbol{a}_{i_1} \bmod \boldsymbol{B}$	$\mathcal{P}(\boldsymbol{B})$	$\tilde{O}(n^4)$	$\tilde{O}(n^3)$
AD$^+$ (12.4)	–	$\boldsymbol{A}$	$\boldsymbol{Ae} + \frac{\mu}{2}\boldsymbol{a}_{i_1} \bmod \boldsymbol{B}$	$\mathcal{P}(\boldsymbol{B})$	$\tilde{O}(n^4)$	$\tilde{O}(n^2)$
R03 (12.7)	$N = 2^{8n^2}$	$\boldsymbol{a}$	$\boldsymbol{ae} + \mu \lfloor a_{i_1}/2 \rceil \bmod N$	$\mathbb{Z}_N$	$\tilde{O}(n^3)$	$\tilde{O}(n^2)$
LWE-PKE (12.8)	$(\boldsymbol{A})$	$\boldsymbol{A}, \boldsymbol{P}$	$(\boldsymbol{Ae}, \boldsymbol{P}^T\boldsymbol{e} + \boldsymbol{\mu}\lfloor q/2 \rfloor)$	$\mathbb{Z}_q^n \times \mathbb{Z}_q^n$	$\tilde{O}(n^2)$	$\tilde{O}(1)$
Dual (12.9)	$(\boldsymbol{A})$	$\boldsymbol{A}, \boldsymbol{U}$	$\boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x}_p, \boldsymbol{U}^T\boldsymbol{s} + \boldsymbol{x}_v + \boldsymbol{\mu}\lfloor q/2 \rfloor$	$\mathbb{Z}_q^m \times \mathbb{Z}_q^n$	$\tilde{O}(n^2)$	$\tilde{O}(1)$

Table 12.1: Comparisons among IND-CPA secure encryption schemes. The factor n denotes the security parameter. See corresponding sections for the details

The other answer is given in 2003, the Regev03 encryption scheme [Reg04b]. He proposed a 1-dimensional version of AD and reduced it security from harder lattice problem than that of Ajtai and Dwork. However, efficiency is not good because the key has huge size. This obstructs us to take a larger security parameter.

These situations were overcome by Regev in 2005 [Reg09] using quantum reductions. He gave a simple public-key encryption scheme based on the variant of Learning Parity Noise (LPN) problem, called Learning With Error (LWE) Problem, and showed the quantum reduction from SIVP_γ to the variant.

On the blowup factor, the ratio between the ciphertext and the plaintext, the original Regev05 encryption scheme has a factor $O(n \log n)$.

Kawachi, Tanaka, and Xagawa [KTX07] reduced the ciphertext blowup factor to $O(n)$. This is dramatically reduced to $O(1)$ by an amortizing technique of Peikert, Vaikuntanathan, and Waters [PVW08].

Gentry, Peikert, and Vaikuntanathan [GPV08] observed that the roles of a public key and a ciphertext can be swapped. This scheme is called as the "Dual" encryption scheme and opens the door to construct identity-based encryption scheme. See [GPV08] and Chapter 14.

The above schemes are secure in the sense of indistinguishability under chosen plaintext attacks (IND-CPA). Can we construct encryption schemes enjoying the stronger security, indistinguishability under chosen ciphertext attacks (IND-CCA2)?

Peikert and Waters [PW08] answered this problem by introducing lossy trapdoor functions and giving the construction of them from LWE problem. Peikert [Pei09c] (and Goldwasser and Vaikuntanathan [GV08]) also gave the answer which exploited the one-wayness of the LWE function, which appears in Chapter 13.

The schemes which we will not mention: Key-leakage security and key-dependent message security are featured recently.

Key-leakage security states the scheme is secure even if the part of the secret key was leaked, which was considered as the theoretical model of side channel attacks. Akavia, Goldwasser, and Vaikuntanathan [AGV09] showed the key-leakage security of the Regev05 encryption scheme. We note that in a weaker model of key-leakage security, the combination of an IND-CPA secure public-key encryption scheme and the extractor yields secure encryption scheme [NS09]. Goldwasser, Kalai, Peikert, and Vaikuntanathan proved the robustness of the LWE based secret-key encryption schemes [GKPV10]. Dodis, Goldwasser, Kalai, Peikert, and Vaikuntanathan [DGK$^+$10] also showed the key-leakage security of the "Dual" encryption scheme in several models. See their papers for the details. We note that the proof techniques of them partially appears in Section 15.4, which are independent.

Key-dependent message security states the scheme is secure even if the ciphertext contains the information of the secret key, for example, $\mathsf{Enc}_{ek}(dk)$ should be indistinguishable from $\mathsf{Enc}_{ek}(0^l)$, where $(ek, dk) \leftarrow \mathsf{KeyGen}(1^n)$ and $l = |dk|$. Formally, the adversary could choose a function from some class $\mathcal{F} = \{f : K_n^k \to M_n\}$, where K_n is a decryption-key space and $M_n = \{0, 1\}^l$ is a message space. The adversary has an access to the oracle which returns $\mathsf{Enc}_{ek_j}(f(sk_1, \ldots, sk_k))$ or $\mathsf{Enc}_{ek_j}(0^l)$ on the query (j, f).

The scheme which satisfies some functions was firstly proposed by Boneh, Halevi, Hamburg, and Ostrovsky [BHHO08]. In the spirit, their scheme has KDM-security with respect to the class of affine functions under the DDH assumption. Applebaum, Cash, Peikert, and Sahai [ACPS09]. also proposed a KDM-secure public-key encryption scheme with respect to the class of affine functions under the LWE assumption. We mention the improvements by Brakerski, Goldwasser, and Kalai [BGK09] and Barak, Haitner, Hofheinz, and Ishai [BHHI09] which proposed the schemes with respect to richer class of functions. For the details see the original papers.

We also does not describe the variants of the Peikert encryption scheme by Katz and Vaikuntanathan [KV09], which yields secure password-based authenticated key-exchange schemes.

Finally, we mention Gentry's fully homomorphic encryption scheme [Gen09] and its variants [SV09, vDGHV09]. Their encryption functions are ring homomorphic to $\mathbb{F}_2$, which solved the long-standing open problem. (We note that Aguilar Melchor, Gaborit, and Herranz [AMGH08] also gave additive homomorphic encryption schemes which allows t multiplications.)

12.2 Definitions

12.2.1 Model of Public-Key Encryption Schemes

A PKE scheme PKE is a quadruplet of algorithms ($\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$).

$\mathsf{Setup}(1^n)$**:** A setup algorithm, given the security parameter 1^n, outputs public parameters *param*.

$\mathsf{KeyGen}(param)$**:** A key-generation algorithm, given *param*, outputs a pair of an encryption key and a decryption key (ek, dk).

$\mathsf{Enc}(param, ek, msg)$**:** An encryption algorithm, given *param*, *ek*, and a message *msg*, outputs a ciphertext *ct*.

$\mathsf{Dec}(dk, ct)$**:** An decryption algorithm, given *dk* and *ct*, returns *msg*.

Correctness: The correctness of a public-key encryption scheme is defined as follows: With overwhelming probability the ciphertext of any message *msg* in the message space under an encryption key *ek* should be decrypted into *msg*, where the probability is taken by coins of Setup, KeyGen, and Enc. Formally, this requirement is denoted

$$\Pr\left[msg \neq \widetilde{msg} : \begin{array}{l} param \leftarrow \mathsf{Setup}(1^n); \\ (ek, dk) \leftarrow \mathsf{KeyGen}(param); \\ ct \leftarrow \mathsf{Enc}(param, ek, msg); \\ \widetilde{msg} \leftarrow \mathsf{Dec}(dk, ct); \end{array} \right] \leq \mathsf{negl}(n).$$

12.2.2 Security Notions

We adopt the standard security notions, indistinguishability of ciphertexts under several attacks. Roughly speaking, any polynomial-time adversary cannot distinguish two ciphertexts of distinct messages *msg* and *msg′* even if it chooses the messages. In chosen plaintext attacks (cpa), the adversary could only encrypt its chosen message and cannot use the decryption oracle. In chosen ciphertext attacks (cca1), the adversary could query to the decryption oracle until the adversary commits the target messages. In chosen ciphertext attacks (cca2), the adversary could query to the decryption oracle after it receives the target ciphertext.

We describe the formal definition as follows: The following experiment is defined in the "second and penalty" style (see [BHK09] for the discussion on the definition styles). Consider the experiment $\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{PKE},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$, where $\text{atk} \in \{\text{cpa}, \text{cca1}, \text{cca2}\}$.

Experiment $\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{PKE},\mathcal{A}}(n)$**:**

Setup Phase: The challenger takes the security parameter n and obtains $param \leftarrow \mathsf{Setup}(1^n)$ and $(ek, dk) \leftarrow \mathsf{KeyGen}(param)$. It gives *param* and *ek* to the adversary $\mathcal{A}$.

Learning Phase 1: The adversary can issue queries to the oracle if $\text{atk} \in \{\text{cca1}, \text{cca2}\}$. The oracle DEC receives an input *ct* and returns $msg \leftarrow \mathsf{Dec}(dk, ct)$.

Challenge Phase: The adversary $\mathcal{A}$ outputs two plaintexts msg_0 and msg_1. The challenger flips a coin $b \leftarrow \{0, 1\}$, sets the target ciphertext to be $ct^* \leftarrow \mathsf{Enc}(ek, msg_b)$, and sends ct^* to the adversary.

Learning Phase 2: The adversary can issue queries to the oracle if atk = cca2. The oracle DEC receives input *ct*. If $ct = ct^*$, the challenger outputs 0 and halts. Otherwise, the oracle returns $msg \leftarrow \mathsf{Dec}(dk, ct)$ to $\mathcal{A}$.

Guessing Phase: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0,1\}$. If $b' = b$, the challenger outputs 1, otherwise 0.

Definition 12.2.1. Let $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{ReKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathsf{PKE},\mathcal{A}}(n) = \left|\Pr\left[\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{PRE},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say that PKE is ind-atk secure if $\mathbf{Adv}^{\text{ind-atk}}_{\mathsf{PKE},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$, where atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$.

12.3 The McEliece Encryption Scheme

As the warm up, we start with a coding-theoretic encryption scheme, the McEliece encryption scheme [McE78] which has appeared in 1978 and is believed that it remains secure against quantum computes. (See the survey [OS08].) The McEliece encryption scheme is as follows (as opposite to ordinal notation, we transpose matrices and rename the parameters):

Scheme 12.3.1 (the McEliece encryption scheme, $\mathsf{McEliece}$)**.** All the participant agree use of an $[m, m-n, 2t+1]$ linear code C, which is able to decode a received word up to t errors.

Key Generation: Let $\boldsymbol{G} \in \mathbb{F}^{n\times m}$ be a generator matrix of the linear code. Then, the public key is $\boldsymbol{G}' = \boldsymbol{\Pi}\boldsymbol{G}\boldsymbol{S}$, where $\boldsymbol{S} \leftarrow \mathrm{GL}_n(\mathbb{F})$ and $\boldsymbol{\Pi}$ is a random permutation matrix over $[m]$. the secret key is $\boldsymbol{S}$, $\boldsymbol{G}$, and $\boldsymbol{\Pi}$.

Encryption: To encrypt the message $\boldsymbol{s} \in \mathbb{F}^k$, choose a error vector $\boldsymbol{x} \leftarrow \mathcal{S}(m,t)$ and compute the ciphertext $\boldsymbol{p} \leftarrow \boldsymbol{G}'^T\boldsymbol{s} + \boldsymbol{x}$.

Decryption: To decrypt it, compute $\boldsymbol{d} = \boldsymbol{\Pi}^{-1}\boldsymbol{p} = \boldsymbol{G}\boldsymbol{S}\boldsymbol{s} + \boldsymbol{\Pi}^{-1}\boldsymbol{x}$, decode it and obtain $\boldsymbol{G}\boldsymbol{S}\boldsymbol{s}$, finally outputs $\boldsymbol{s}$.

We already see $\mathcal{S}(m,t)$ is an enumeration set. Hence, for any vector $\boldsymbol{x} \in \mathcal{S}(m,t)$, $\pi(\boldsymbol{x})$ is also in $\mathcal{S}(m,t)$. Thus, the decoding procedure works well in decryption.

The security is guaranteed from the intuition that one could not find the structure $\boldsymbol{G}$ from $\boldsymbol{G}'$ in polynomial time of n. The one-wayness follows from the two assumptions; the one is that that the learning with parity (LPN) problem is hard and the other is the indistinguishability of the public key and the uniform over $\mathbb{F}^{n\times m}$.

We will turn back to this encryption scheme, because this and some lattice-based encryption schemes share some structures.

12.4 The Ajtai–Dwork Encryption Scheme

Ajtai and Dwork proposed three public-key encryption scheme based on lattice problems. The third of them is well-known as the Ajtai–Dwork encryption scheme.

The first construction of lattice-based public-key encryption scheme is the Ajtai–Dwork encryption scheme [AD97]. (The first and the second are preparations for the third cryptosystem.) Later, Goldreich, Goldwasser, and Halevi [GGH97a] eliminated decryption errors in the public-key encryption scheme. The scheme enjoys the average-case/worst-case security proof and is based on uSVP with an approximation factor $\tilde{O}(n^{11})$. The scheme is later improved by the originators, Ajtai and Dwork.

We give a rough structure of the encryption scheme. The secret key is a unit n-dimensional vector $\boldsymbol{u} \in B_n(1)$. Imagine the set of the hyperplanes $H = \{\boldsymbol{x} \in \mathbb{R}^n \mid \langle \boldsymbol{x}, \boldsymbol{u} \rangle \in \mathbb{Z}\}$. Then, the public key consists of m vectors near H. The ciphertext of 0, $\boldsymbol{c}_0$, is the random sum of such m vectors, which is also near H. The ciphertext of 1, $\boldsymbol{c}_1$, is a random vector $\mathbb{R}^n$. One knowing $\boldsymbol{u}$ can distinguish the ciphertext of 0 from the ciphertext of 1 by determining $\langle \boldsymbol{c}, \boldsymbol{u} \rangle$ is near H or not.

12.4.1 Description

For $d \in \mathbb{R}$, let $\mathrm{frc}(d)$ denote $|d - \lfloor d \rceil|$, which stands for the distance of d from the integer set $\mathbb{Z}$.

The scheme AD is described as follows:

Scheme 12.4.1 (AD [AD97])**.** All the participants agree with the parameters n, $m = n^3$, $R = 2^{O(n \log n)}$, and $r = n^{-3}$.

$\mathsf{Setup}(1^n)$**:** Given the security parameter n output $\perp$.

$\mathsf{KeyGen}(\mathit{param})$**:** Choose $\boldsymbol{u} \leftarrow B(1)$. Choose $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \leftarrow \{\boldsymbol{x} \in B(R) \mid \langle \boldsymbol{x}, \boldsymbol{u} \rangle \in \mathbb{Z}\}$. Choose $\boldsymbol{y}_{i,j} \leftarrow B(r)$ uniformly at random for $i = 1, \dots, m$ and $j = 1, \dots, n$. Compute $z_i = \sum_{j=1}^{n} \boldsymbol{y}_{i,j}$ for $i = 1, \dots, m$. Then, compute $\boldsymbol{a}_i = \boldsymbol{x}_i + z_i$. Let i_0 be the smallest i for which the width of parallelepiped spanned by $\boldsymbol{a}_{i+1}, \dots, \boldsymbol{a}_{i+n}$ is at least n^{-2}. For $j = 1, \dots, n$, let $\boldsymbol{b}_j = \boldsymbol{a}_{i_0+j}$. The decryption key is $dk = \boldsymbol{u}$ and the encryption key is $ek = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_m, i_0)$.

$\mathsf{Enc}(ek = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_m, i_0), msg = t)$**:** Let $t \in \{0, 1\}$ be a plaintext.

- To encrypt $t = 0$, choose $\boldsymbol{e} \in \{0, 1\}^m$ and compute $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{e} \bmod \boldsymbol{B}$, where $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$.
- To encrypt $t = 1$, choose $\boldsymbol{c} \leftarrow \mathcal{P}(\boldsymbol{B})$. The ciphertext is $\boldsymbol{c}$.

$\mathsf{Dec}(dk = \boldsymbol{u}, ct = \boldsymbol{c})$**:** For a received ciphertext, compute $d = \langle \boldsymbol{c}, \boldsymbol{u} \rangle$. Output 0 if $|\mathrm{frc}(d)| \leq 1/n$, 1 otherwise.

Obviously, the ciphertext of 1 is decrypted as 0 with probability about $2/n$. In order to eliminate the decryption errors in AD, Goldreich, Goldwasser, and Halevi [GGH97a] changed several procedures. We denote the scheme by $\mathsf{AD\text{-}GGH}$.

Scheme 12.4.2 ($\mathsf{AD\text{-}GGH}$ [GGH97a])**.** All the participants agree with the parameters n, $m = n^3$, $R = 2^{O(n \log n)}$, and $r = n^{-3}$.

$\mathsf{Setup}(1^n)$**:** The same as $\mathsf{AD.Setup}$

KeyGen($param$)**:** It chooses $\boldsymbol{u}$, $\boldsymbol{x}_i$, and z_i, computes $\boldsymbol{a}_i = \boldsymbol{x}_i + z_i$, selects i_0 in the same manner as AD.KeyGen does. In addition, pick an index i_1 uniformly at random from all indices i for which $\langle \boldsymbol{a}_i, \boldsymbol{u} \rangle$ is odd. The decryption key is $dk = \boldsymbol{u}$ and the encryption key is $ek = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_m, i_0, i_1)$.

Enc($ek = (\boldsymbol{a}_1, \dots, \boldsymbol{a}_m, i_0, i_1), msg = t$)**:** Let $t \in \{0, 1\}$ be a plaintext. Choose $\boldsymbol{e} \in \{0, 1\}^m$ and compute $\boldsymbol{c} = \boldsymbol{A}\boldsymbol{e} + t \cdot \boldsymbol{a}_{i_1}/2 \bmod \boldsymbol{B}$, where $\boldsymbol{B} = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_n]$.

Dec($dk = \boldsymbol{u}, ct = \boldsymbol{c}$)**:** For a received ciphertext, compute $d = \langle \boldsymbol{c}, \boldsymbol{u} \rangle$. Output 0 if $|\mathrm{frc}\,(d)| \leq 1/4$, 1 otherwise.

12.4.2 Security

The security of AD and AD-GGH is based on $O(n^{11})$-uSVP.

Consider the following two games: The first game is the original IND-CPA game. In the second game, we replace the public key with the uniformly chosen one, that is, $\boldsymbol{a}_i \leftarrow B(R)$. Roughly speaking, if the adversary distinguishes two games, we can verify a vector $\boldsymbol{p}$ is near to the set of hyperplanes $H = \{\boldsymbol{x} \in \mathbb{R}^n \mid \langle \boldsymbol{x}, \boldsymbol{u} \rangle \in \mathbb{Z}\}$ or not. Exploiting this power, the security is reduced from uSVP with approximation factor $\gamma = O(n^{11})$. For the detailed proof, see the original paper [AD97].

12.4.3 Attacks

The scheme AD has a fatal drawback in the real world: huge public key of bit size $\tilde{O}(n^4)$. The size of the public key is approximately 2Gb even when $n = 32$. Nguyen and Stern [NS98] analyzed the scheme with realistic parameters by the LLL algorithm.

Hall, Goldberg, and Schneier [HGS99] examined the CCA1 attacks against several public-key encryption schemes based on combinatorial problems including AD-GGH. They proposed the CCA1 attacks which retrieves the secret key of AD-GGH. Izmerly and Mor [IM06] gave the CCA1 attacks against AD and AD-GGH.

These two attacks employs the above idea in the security reduction. If one has a decryption oracle, one recognize the set of hyperplanes H. Using the set of hyperplanes, one can extract the secret key $\boldsymbol{u}$.

12.5 The Goldreich–Goldwasser–Halevi Encryption Scheme

At the same time of the improvement of AD [GGH97a], Goldreich, Goldwasser, and Halevi proposed their new encryption scheme GGH. Their sales point is significantly small size of the public key $O(n^3)$ (still larger than conservative encryption schemes).

(We note that Solé, Charnes, and Martin rediscover GGH as the lattice version of the McEliece encryption scheme. See [SCM01].)

Roughly speaking, their idea is as follows: The secret key is a "good" basis $\boldsymbol{R}$ of a random lattice L, which corresponds to $\boldsymbol{G}$. The public key is a "bad" basis $\boldsymbol{B} = \boldsymbol{R}\boldsymbol{U}^{-1}$ of the random lattice, where $\boldsymbol{U}$ is a unimodular matrix and $\boldsymbol{B}$ corresponds to $\boldsymbol{G}'$. The message is encoded into the coefficient vector $\boldsymbol{s}$. Then, the ciphertext is $\boldsymbol{p} = \boldsymbol{B}\boldsymbol{s} + \boldsymbol{x}$ where $\boldsymbol{x}$ is a small random error vector, which resembles to the ciphertext of the McEliece encryption scheme. To decrypt $\boldsymbol{p}$, first apply the round-off algorithm, that is, $\boldsymbol{d} \leftarrow \lfloor \boldsymbol{R}^{-1}\boldsymbol{p} \rceil$. Then, $\boldsymbol{d}$ will be $\boldsymbol{R}^{-1}\boldsymbol{B}\boldsymbol{s}$ since the error $\boldsymbol{x}$ is short and $\boldsymbol{R}$ is good. Multiplying $\boldsymbol{B}^{-1}\boldsymbol{R}$ to $\boldsymbol{d}$, we can obtain $\boldsymbol{s}$.

Why it works: The inversion algorithm computes $\boldsymbol{s} = \boldsymbol{U}\lfloor \boldsymbol{R}^{-1}\boldsymbol{p} \rceil$ and $\boldsymbol{x} = \boldsymbol{p} - \boldsymbol{B}\boldsymbol{s}$, where $\boldsymbol{U} = \boldsymbol{B}^{-1}\boldsymbol{R}$. Hence, we should consider $\lfloor \boldsymbol{R}^{-1}\boldsymbol{p} \rceil$, because $\boldsymbol{x}$ is computed automatically from $\boldsymbol{s}$. We have that

$$\lfloor \boldsymbol{R}^{-1}\boldsymbol{p} \rceil = \lfloor \boldsymbol{R}^{-1}(\boldsymbol{B}\boldsymbol{s} + \boldsymbol{x}) \rceil = \lfloor \boldsymbol{U}^{-1}\boldsymbol{s} + \boldsymbol{R}^{-1}\boldsymbol{x} \rceil.$$

Since $\boldsymbol{U}$ is unimodular, $\boldsymbol{U}^{-1}$ is also unimodular. Thus, the computation correctly works when $\lfloor \boldsymbol{R}^{-1}\boldsymbol{x} \rceil = \boldsymbol{0}$, that is, $\left\|\boldsymbol{R}^{-1}\boldsymbol{x}\right\|_\infty < 1/2$.

Observe that $\boldsymbol{R}^{-1} = (\boldsymbol{R}^{-T})^T$. Then, we only need to show that a basis $\boldsymbol{T} = \boldsymbol{R}^{-T}$ of the dual lattice Λ^* is short. If $\boldsymbol{T}$ is short, we have the following lemma ensuring the correctness.

Lemma 12.5.1. *Let $\boldsymbol{B}$ be a basis of an n-dimensional lattice Λ. Let $\boldsymbol{T}$ be a basis of a dual lattice Λ^* such that $\|\boldsymbol{T}\| \le L$. For any $\boldsymbol{s} \in \mathbb{Z}^n$ and any $\boldsymbol{x} \in \mathbb{R}^n$ with $\|\boldsymbol{x}\| \le 1/2L$, we have*

$$\lfloor \boldsymbol{T}^T(\boldsymbol{B}\boldsymbol{s} + \boldsymbol{x}) \rceil = \boldsymbol{T}^T\boldsymbol{B}\boldsymbol{s}.$$

Proof. Since $\boldsymbol{T} = [\boldsymbol{t}_1, \dots, \boldsymbol{t}_n]$ is a basis of the dual lattice Λ^*, we have $\langle \boldsymbol{t}_i, \boldsymbol{B}\boldsymbol{s} \rangle \in \mathbb{Z}$ for any $i \in [n]$ and $\boldsymbol{s} \in \mathbb{Z}^n$. Thus, $\boldsymbol{T}^T\boldsymbol{B}\boldsymbol{s} = (\langle \boldsymbol{t}_1, \boldsymbol{B}\boldsymbol{s} \rangle, \dots, \langle \boldsymbol{t}_n, \boldsymbol{B}\boldsymbol{s} \rangle) \in \mathbb{Z}^n$. Hence, we only need to show that $\left\|\boldsymbol{T}^T \cdot \boldsymbol{x}\right\|_\infty < 1/2$, that is, for any $i \in [n]$, $|\langle \boldsymbol{t}_i, \boldsymbol{x} \rangle| < 1/2$.

By the hypothesis of $\boldsymbol{T}$, we have $\|\boldsymbol{t}_i\| \le L$. Therefore, we have that

$$|\langle \boldsymbol{t}_i, \boldsymbol{x} \rangle| \le \|\boldsymbol{t}_i\| \cdot \|\boldsymbol{x}\| < L \cdot (1/2L) = 1/2,$$

which completes the proof. □

12.5.1 Description

The key-generation algorithms vary and the authors defined two key-generation algorithms:

1. KeyGen1: This algorithm chooses a "random" lattice: Generate $\boldsymbol{R} \leftarrow \{-l, -(l-1), \dots, l-1, l\}^{n \times n}$, where l is a small integer, say 4.

2. KeyGen2: This algorithm chooses a "rectangular" lattice: Generate noise matrix $\boldsymbol{R}' \leftarrow \{-l, \dots, l\}^{n \times n}$, where l is a small integer, say 4. Then, compute $\boldsymbol{R} \leftarrow \boldsymbol{R}' + k\boldsymbol{I}_n$, where k is a large integer, say $\sqrt{n}$.

Scheme 12.5.2 (GGH)**.**

KeyGen(1^n)**:** It outputs $\boldsymbol{B}$ and $\boldsymbol{R}$, which spans the same lattice, by using KeyGen1 or KeyGen2.

Enc($\boldsymbol{B}, \boldsymbol{s}$)**:** It first choose $\boldsymbol{x} \leftarrow D_n$, where $D_n \subseteq \mathbb{Z}^n$ and each element in D_n is short. Then, the ciphertext is $\boldsymbol{p} = \boldsymbol{B}\boldsymbol{s} + \boldsymbol{x}$.

Dec($\boldsymbol{R}, \boldsymbol{p}$)**:** The decryption algorithm computes $\boldsymbol{s} = \boldsymbol{B}^{-1}\boldsymbol{R}\left\lfloor \boldsymbol{R}^{-1}\boldsymbol{p} \right\rceil$ and outputs it.

12.5.2 Attacks

The scheme GGH has no security proofs. Nguyen [Ngu99] reported a weak point of GGH: the error vector $\boldsymbol{x}$ is chosen from $D_n = \{-\sigma, \sigma\}^n$. He solved the challenge by the authors of GGH up to $n = 350$ and gave the partial solution even for $n = 400$. Hence, the error vector should be chosen from $D_n = \{-\sigma, -(\sigma - 1), \dots, \sigma - 1, \sigma\}$.

After several years, Lee and Hahn [LH08] solved the GGH challenge of parameter $n = 400$ completely using Nguyen's partial solution. This demonstrates the security parameter n must be large enough.

12.5.3 Micciancio's Variant

In 2001, Micciancio [Mic01] gives the dual of the scheme. We give the details of the scheme.

Scheme 12.5.3 (GGH-Mic)**.**

KeyGen(1^n)**:** It first generates $\boldsymbol{R}$ as in KeyGen1 or KeyGen2. It next computes an Hermite normal form $\boldsymbol{B}$ of $\boldsymbol{R}$. Then, the public key is $\boldsymbol{B}$ and the secret key is $\boldsymbol{R}$.

Enc($\boldsymbol{B}, \boldsymbol{x}$)**:** The message vector is $\boldsymbol{x} \in D_n$. Then, the ciphertext is $\boldsymbol{u} = \boldsymbol{x} \bmod \boldsymbol{B}$.

Dec($\boldsymbol{R}, \boldsymbol{u}$)**:** Applying the nearest plane algorithm to $\boldsymbol{u}$ with $\boldsymbol{R}$, it obtains the closest vector $\boldsymbol{R}\boldsymbol{s}$ to the vector $\boldsymbol{u}$. Then output $\boldsymbol{x} = \boldsymbol{R}\boldsymbol{s} - \boldsymbol{u}$.

We note that the scheme can be considered as a lattice analogue of the Niederreider encryption scheme. Micciancio's main idea is use of Hermite normal forms (HNFs). The benefits derived from the HNFs are simplifying the randomizing method and reducing the sizes of the public keys and the ciphertexts. First, since the Hermite normal form of the basis is computed deterministically, we can ignore the "bad" randomness of the public keys. Second, the Hermite normal form is upper triangle and its non-diagonal elements are smaller than the corresponding diagonals, whose product is at most $2^{O(n \log n)}$. Hence the size of ciphertext is now $O(n \log n)$.

We finally mention the improvement by Plantard, Rose, and Susilo, which speeds up the decryption procedure of the cryptosystem. See [PRS09] for the detailed analyses and the experimental results.

12.5.4 The Variant by Paeng, Jung, and Ha

Paeng, Jung, and Ha [PJH03] proposed the technique to reduce the size of public key. This very resembles NTRU detailed in later. Let us fix $n' = 2n$ and consider $R_q = \mathbb{Z}_q[x]/\langle x^n - 1\rangle$.

Their key generation algorithm first choose four polynomials $\mathbf{f}_1, \mathbf{f}_2, \mathbf{h}_1, \mathbf{h}_2 \in R_q$. The polynomials $\mathbf{f}_i$ has a large coefficient $\lfloor\sqrt{2nl}\rfloor$ in some position j_i and the other coefficients are chosen from $[-l, l]$. The polynomials $\mathbf{h}_i$ are chosen from $[-l, l]^n$. Then, the private matrix $\boldsymbol{R} = \mathrm{Rot}_{x^n-1}(\boldsymbol{R}')$ is

$$\boldsymbol{R}' = \begin{bmatrix} \mathbf{f}_1 & \mathbf{h}_1 \\ \mathbf{h}_2 & \mathbf{f}_2 \end{bmatrix}.$$

If the positions $j_i = 1$, then this matrix is in the range of the KeyGen2.

In order to randomize the matrix, they are chosen $\mathbf{g} \in (-q/2, q/2]^n$, which has the inverse $\mathbf{g}^{-1}$ in R_q. This shows that there exist $\mathbf{g}_q$ and $\mathbf{Q}$ in $R = \mathbb{Z}[x]/\langle x^n - 1\rangle$ such that $\mathbf{g} \otimes \mathbf{g}_q - 1 = q\mathbf{Q}$ over $\mathbb{Z}$. They computed four polynomials $\mathbf{p}_i$ as follows:

$$\boldsymbol{B}' = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 \\ \mathbf{p}_3 & \mathbf{p}_4 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{f}_1 & \mathbf{h}_1 \\ \mathbf{h}_2 & \mathbf{f}_2 \end{bmatrix}}_{\boldsymbol{R}'} \otimes \underbrace{\begin{bmatrix} \mathbf{g} & q \\ \mathbf{Q} & \mathbf{g}_q \end{bmatrix}}_{\boldsymbol{U}^{-1}}.$$

It is easy to verify $\mathrm{Rot}_{x^n-1}(\boldsymbol{U}^{-1})$ has a determinant 1, since $\mathbf{g} \times \mathbf{g}_q - q\mathbf{Q} = 1$ over $\mathbb{Z}$.

They replaced the key-generation algorithm in GGH with the above and $\boldsymbol{R} = \mathrm{Rot}_{x^n-1}(\boldsymbol{R}')$ and $\boldsymbol{B} = \mathrm{Rot}_{x^n-1}(\boldsymbol{B}')$ as the secret and the public key. This drastically reduce the length of key, the size of $\boldsymbol{B}$ is 22.3kB even if we set $n = 1001$ and take 80-bit prime q.

Unfortunately, after four years, Han, Kim, and Yeom [HKY07] analyzed the PJH variant up to $n = 1001$. They reported they can recover the secret key $\boldsymbol{R}'$ from $\boldsymbol{B}'$ within 10 minutes computations. They observed that $\mathbf{g} \otimes \mathbf{P}_2 = q\mathbf{P}_1 + \mathbf{h}_1$ over $\mathbb{Z}$. This indicates the total system is broken if $\mathbf{h}_1$ and q are recovered, since we can find $\mathbf{g}$ from $\mathbf{h}_1$ and q by the above equation and also other variables. In addition, the lattice spanned by $\mathrm{Rot}_{x^n-1}(\mathbf{p}_2)$ has very short vector $\mathbf{h}_1$ and other vectors in the lattice will very long as q. Using this property, they recovered $\mathbf{h}_1$ and q from $\mathbf{p}_2$ heuristically. For the details, see the original paper [HKY07].

12.6 NTRU

Although we already introduced this encryption scheme in Chapter 9, we review it in the context of the lattice-based encryption schemes. See the introduction of NTRU in Section 9.1.

12.6.1 Description

For details, see the original paper [HPS98] and the proposals of the parameters [HS00, HGSW05, HHGP+07, WHGH+08, HHHGW09].

For a positive integer n which is often set as a prime, NTRU is defined on a quotient ring $R = \mathbb{Z}[x]/\langle x^n - 1\rangle$. For a positive integer or a small polynomial q, we denote $\mathbb{Z}[x]/\langle q, x^n - 1\rangle$ by R_q.

Intuitively, the security is based on the hardness to factor a product of two short polynomials in R_q.

Scheme 12.6.1 (NTRUEncrypt)**.** Let n denote the dimension of R_q. The subsets of R_q, $\mathcal{L}_f$, $\mathcal{L}_g$, $\mathcal{L}_m$, $\mathcal{L}_r$, and $\mathcal{L}_F$ are defined later. They are used for key generation, encryption, and decryption.

Setup(1^n)**:** Given the security parameter 1^n, output 1^n.

KeyGen($param = 1^n$)**:** Choose $\mathbf{f} \leftarrow \mathcal{L}_f$ and $\mathbf{g} \leftarrow \mathcal{L}_g$ with the constrain that $\mathbf{f}$ is invertible in R_q and R_p. Set $\mathbf{F}_q \leftarrow \mathbf{f}^{-1}$ in R_q. Compute $\mathbf{h} \leftarrow p \otimes \mathbf{g} \otimes \mathbf{F}_q$ in R_q. The public key is $\mathbf{h}$ and the secret key is $\mathbf{f}$.

Enc($ek = \mathbf{h}, msg = \mathbf{m}$)**:** The plaintext is $\mathbf{m} \in \mathcal{L}_m$. Generate a random polynomial $\mathbf{r} \leftarrow \mathcal{L}_r$ and compute $\mathbf{c} \leftarrow \mathbf{h} \otimes \mathbf{r} + \mathbf{m}$ in R_q. The ciphertext is $\mathbf{c}$.

Dec($dk = \mathbf{f}, ct = \mathbf{c}$)**:** The ciphertext is $\mathbf{c} \in R_q$. Compute $\mathbf{a}' \leftarrow \mathbf{f} \otimes \mathbf{c}$ in R_q. Compute $\mathbf{a} \leftarrow p \otimes \mathbf{g} \otimes \mathbf{r} + \mathbf{f} \otimes \mathbf{m}$ in R from $\mathbf{a}'$ by using a centering algorithm. Compute $\mathbf{F}_p \leftarrow \mathbf{f}^{-1}$ in R_p. Compute $\mathbf{m}' \leftarrow \mathbf{F}_p \otimes \mathbf{a}$ in R_p. The obtained plaintext is $\mathbf{m}'$.

The decryption correctly works since the parameters are chosen carefully to ensure that $\mathbf{a} = p \otimes \mathbf{g} \otimes \mathbf{r} + \mathbf{f} \otimes \mathbf{m}$ in R with high probability. We omit the details of the parameter setting; see the original paper or the papers on instantiations [HPS98, WHGH+08, HHHGW09].

Let $\mathcal{T}$ denote $\{-1, 0, +1\}^n$. $\mathcal{T}(d_1, d_2)$ denotes the subset of $\mathcal{T}$ such that each polynomial in $\mathcal{T}(d_1, d_2)$ has exactly d_1 coefficients set to 1 and d_2 coefficients set to -1. For an integer a and a subset $\mathcal{S} \subseteq R_q$, we define $a\mathcal{S}$ as $\{a\mathbf{f} : \mathbf{f} \in \mathcal{S}\}$. For a subset $\mathcal{S} \subseteq R_q$, $\mathcal{S}^*$ denotes the set of the polynomials in $\mathcal{S}$ which have the inverses in R_q, i.e., $\mathcal{S}^* = \{\mathbf{f} \in \mathcal{S} : \exists \mathbf{f}^{-1} \in R_q\}$.

There are five instantiations of NTRU, NTRU-1998 [HPS98], NTRU-2001 [HS00], NTRU-2005 [HGSW05], NTRU-2007 [HHGP+07], and NTRU-2008 [WHGH+08, HHHGW09]. For simplicity, we only consider NTRU-1998 and NTRU-2008 in this paper. The following table summarizes the parameter sets of these instantiations.

Parameter Sets	q	p	$\mathcal{L}_f$	$\mathcal{L}_g$	$\mathcal{L}_m$	$\mathcal{L}_r$	$\mathcal{L}_F$
NTRU-1998	2^k	3	$\mathcal{T}(d_f, d_f - 1)^*$	$\mathcal{T}(d_g, d_g)$	$\mathcal{T}$	$\mathcal{T}(d_r, d_r)$	-
NTRU-2001	prime	$2 + \alpha$	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{B}(d_g)$	$\mathcal{B}$	$\mathcal{B}(d_r)$	$\mathcal{B}(d_F)$
NTRU-2005	prime	2	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{B}(N/2)^*$	$\mathcal{B}$	$\mathcal{X}(d_r)$	$\mathcal{X}(d_F)$
NTRU-2007	2^k	3	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{T}(d_f, d_f - 1)^*$	$\mathcal{T}(d_f, d_f - 1)$	$\mathcal{T}(d_f, d_f - 1)$	$\mathcal{T}(d_f, d_f - 1)$
NTRU-2008	2^k	3	$\{1 + p \otimes \mathbf{F} : \mathbf{F} \in \mathcal{L}_F\}^*$	$\mathcal{T}(d_g, d_g)$	$\mathcal{T}$	$\mathcal{T}(d_r, d_r)$	$\mathcal{T}(d_F)$

Interpretation as the Micciancio variant of the GGH encryption: Here, we note an interpretation in Micciancio and Goldwasser [MG02] and Micciancio and Regev [MR08]. We have already defined the NTRU lattice [CS97] in Section 9.3. For a secret key $(\mathbf{f}, \mathbf{g})$ and a public key $\mathbf{h}$, the NTRU lattice $\Lambda_{\mathbf{h}}$ is defined as

$$\Lambda_{\mathbf{h}} = \mathcal{L}(\boldsymbol{H}) = \Lambda_q(\boldsymbol{C}) = \Lambda_q^{\perp}(\boldsymbol{A}),$$

where

$$\begin{aligned}
\boldsymbol{H} &= \begin{bmatrix} \mathrm{Rot}_{x^n-1}(1) & \mathrm{Rot}_{x^n-1}(0) \\ \mathrm{Rot}_{x^n-1}(\mathbf{h}) & \mathrm{Rot}_{x^n-1}(q) \end{bmatrix}, \\
\boldsymbol{C} &= \begin{bmatrix} \mathrm{Rot}^T_{x^n-1}(\mathbf{f}) & \mathrm{Rot}^T_{x^n-1}(p \otimes \mathbf{g}) \end{bmatrix}, \\
\boldsymbol{A} &= \begin{bmatrix} -\mathrm{Rot}(\mathbf{h}) & \mathrm{Rot}(1) \end{bmatrix}.
\end{aligned}$$

Notice that $\boldsymbol{H}$ is indeed an Hermite normal form because $\mathbf{h} \in [0, q-1]^n$. Notice also that $\Lambda_{\mathbf{h}}$ contains As the consequence, it contains the short vector $(\boldsymbol{f}, p\boldsymbol{g})$, since $-\mathbf{f} \otimes \mathbf{h} + p\mathbf{g} \equiv 0 \pmod{q}$.

Consider the vector $(-\mathbf{r}, \mathbf{m})$ and reduce it modulo $\boldsymbol{H}$.

$$\begin{bmatrix} -\mathbf{r} \\ \mathbf{m} \end{bmatrix} \bmod \begin{bmatrix} \mathrm{Rot}_{x^n-1}(1) & \mathrm{Rot}_{x^n-1}(0) \\ \mathrm{Rot}_{x^n-1}(\mathbf{h}) & \mathrm{Rot}_{x^n-1}(q) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{m} + \mathbf{h} \otimes \mathbf{r} \bmod q \end{bmatrix}.$$

This indicates the encryption procedure corresponds to that of Micciancio's variant.

12.6.2 Attacks

In the next year, Coppersmith and Shamir [CS97] proposed a lattice-based attack against NTRU using the above notions. They studied NTRU lattices and observed that a secret key comprises of a half of a short basis of the NTRU lattice which is generated by the correspondent public key of the cryptosystem. They also observed that a ciphertext is the remainder of the concatenation of a message vector and a random vector modulo the NTRU lattice. May [May99], , see Jaulmes and Joux [JJ00], Han, Hong, Han, and Kwon [HHHK03], Howgrave-Graham, Nguyen, Pointcheval, Proos, Silverman, Singer, and Whyte [HGNP+03], Meskanen and Renvall [MR06], and Gama and Nguyen [GN07].

12.7 The Regev03 Encryption Scheme

Regev [Reg04b] improved the Ajtai-Dwork public-key encryption scheme AD. The underlying assumption is the worst-case hardness of uSVP with a factor $\tilde{O}(n^{1.5})$. We can consider the scheme as a 1-dimensional version of AD-GGH.

12.7.1 Description

Scheme 12.7.1 (mR03 [Reg04b, KTX07])**.** All the participants agree with the parameters n, r, and $\delta(n) = \omega(n^{1+r}\sqrt{\log n})$, the precision 2^{-8n^2}, and the size p of the plaintext space. We define $H_r = \{h \in [\sqrt{N}, 2\sqrt{N}) : \mathrm{frc}(h) < 1/(8n^r m)\}$.

Setup(1^n)**:** Given the security parameter 1^n, output 1^n.

KeyGen(1^n)**:** We choose $h \in H_r$ uniformly at random and set $d = N/h$. Choosing $\alpha \in [2/\delta(n), (2\sqrt{2})/\delta(n))$, we sample m values $z_1, \ldots, z_m$ from the distribution $\Phi_{h,\alpha}$, where $z_i = (x_i + y_i)/h$ $(i = 1, \ldots, m)$ according to the above sampling procedure. Let $a_i = \lfloor N z_i \rfloor$ for every $i \in \{1, \ldots, m\}$. Additionally, we choose an index i_1 uniformly at random from $\{i : x_i \not\equiv 0 \bmod p\}$. Then, we compute $k \equiv x_j \bmod p$. The decryption key is $dk = (d, k)$ and the encryption key is $ek = (a_1, \ldots, a_m, i_1)$.

Enc($ek = (a_1, \ldots, a_m, i_1), msg = t$)**:** Let $\sigma \in \{0, \ldots, p-1\}$ be a plaintext. We choose a uniformly random subset S of $\{1, \ldots, m\}$. The ciphertext is $w = (\sum_{i \in S} a_i + t\lfloor a_{i_1}/p \rceil) \bmod N$.

Dec($dk = (d, k), ct = w$)**:** For a received ciphertext $w \in \{0, \ldots, N-1\}$, we compute $\tau = w/d \bmod 1$. We decrypt the ciphertext w to $\lfloor p\tau \rceil k^{-1} \bmod p$, where k^{-1} is the inverse of k in $\mathbb{Z}_p$.

12.7.2 Security and Attacks

The security is summarized as follows:

Theorem 12.7.2. *For any constant $r > 0$, let $\delta(n) = \omega(n^{1+r}\sqrt{\log n})$ and let $p(n)$ be a prime such that $2 \le p(n) \le n^r$. The cryptosystem* mR03 *encrypts a $\lfloor \log p(n) \rfloor$-bit plaintext into an $8n^2$-bit ciphertext with decryption error probability at most $2^{-\Omega(\delta^2(n)/(n^{2r}m))} + 2^{-\Omega(n)}$. The security of* mR03 *is based on the worst case of $O(\delta(n)\sqrt{n})$-uSVP.*

On the attacks, we found only Izmerly and Mor [IM06] gave the CCA1 attack against the original scheme R03.

12.8 The Regev05 Encryption Scheme

In 2005, Regev [Reg09] proposed a lattice-based public-key encryption scheme based on the LWE problem. Formally, the dLWE(q,χ) assumption is defined as follows:

Definition 12.8.1 (dLWE assumption)**.** For $q = q(n)$, a distribution χ, and an adversary $\mathcal{A}$, define the advantage of the adversary as follows:

$$\mathbf{Adv}_{\mathrm{dLWE}(q,\chi),\mathcal{A}}(n) = |\Pr[\mathcal{A}^{A_{s,\chi}}(1^n) = 1] - \Pr[\mathcal{A}^{U}(1^n) = 1]|,$$

where probability is taken by $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and random coins of $\mathcal{A}$. We say that the dLWE(q,χ) assumption holds if for any polynomial-time adversary, $\mathbf{Adv}_{\mathrm{dLWE}(q,\chi),\mathcal{A}}(n)$ is negligible in n.

Before reviewing LWE-PKE, we consider the following simple symmetric-key encryption scheme LWE-SKE based on the dLWE(q,χ) assumption.

Enc($dk = \boldsymbol{s} \in \mathbb{Z}_q^n, msg = w \in \{0,1\}$)**:** It generates $\boldsymbol{a} \leftarrow \mathbb{Z}_q^n$ and $x \leftarrow \chi$ randomly, and outputs $(\boldsymbol{a}, v = \langle \boldsymbol{a}, \boldsymbol{s}\rangle + \chi + w\lfloor q/2\rceil)$.

Dec($dk = \boldsymbol{s}, ct = (\boldsymbol{a}, v)$)**:** It computes $d = v - \langle \boldsymbol{a}, \boldsymbol{s}\rangle$ and outputs 0 if $|d|_q \leq q/4$ and 1 otherwise.

Notice that the ciphertext of 0 is the sample from $A_{\boldsymbol{s},\chi}$. One cannot distinguish the ciphertexts of 0 and 1 if the dLWE(q,χ) assumption holds.

Regev's encryption scheme, R05, is obtained from LWE-SKE as follows: The public key is m ciphertexts of 0. The ciphertext of R05 is the random sum of the ciphertexts plus $w\lfloor q/2\rceil$. Since LWE-SKE is bounded homomorphic, the decryption works correctly.

The security proof is done by as follows: The public key (m samples from $A_{\boldsymbol{s},\chi}$) and the uniform distribution are computationally indistinguishable. Hence, we can replace the public key and the random vectors over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. In addition, the random sum of the random vectors is almost uniformly distributed by the leftover hash lemma. Therefore, the adversary cannot distinguish the ciphertexts of two messages after the replacement.

12.8.1 Description

Scheme 12.8.2 (LWE-PKE [Reg09])**.** Define the function $t(a) = \lfloor aq/p\rceil \bmod q$ for $a \in \mathbb{Z}_p$. Naturally, for the vector $\boldsymbol{a} = (a_1, \ldots, a_l) \in \mathbb{Z}_p^l$, we define $t(\boldsymbol{a}) = (t(a_1), \ldots, t(a_l)) \in \mathbb{Z}_q^l$.

Setup(1^n)**:** On input the security parameter n, it outputs the random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ as *param*.

KeyGen($param = \boldsymbol{A}$)**:** It generates $\boldsymbol{S} \leftarrow \mathbb{Z}_q^{n\times l}$ and $\boldsymbol{X} \leftarrow \chi^{m\times l}$. It outputs $\boldsymbol{P} = \boldsymbol{A}^T\boldsymbol{S} + \boldsymbol{X} \in \mathbb{Z}_q^{m\times l}$.

Enc($param = \boldsymbol{A}, ek = \boldsymbol{P}, \boldsymbol{b}$)**:** For message $\boldsymbol{v} \in \mathbb{Z}_p^l$, define the new vector $\boldsymbol{w} = t(\boldsymbol{b}) \in \mathbb{Z}_p^l$. Choose a vector $\boldsymbol{e} \leftarrow \{0,1\}^m \subset \mathbb{Z}_q^m$ uniformly at random. The ciphertext is the pair $(\boldsymbol{u}, \boldsymbol{c}) = (\boldsymbol{A}\boldsymbol{e}, \boldsymbol{P}^T\boldsymbol{e} + \boldsymbol{w}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$.

Dec($dk = \boldsymbol{S}, ct = (\boldsymbol{u}, \boldsymbol{c})$) . Compute $\boldsymbol{d} = \boldsymbol{c} - \boldsymbol{S}^T\boldsymbol{u} \in \mathbb{Z}_q^l$. Output the plaintext $\boldsymbol{b} \in \mathbb{Z}_p^l$ with $d_i - t(b_i) \in \mathbb{Z}_q$ is closest to 0.

Theorem 12.8.3 (Correctness, [PVW08, Lemma 7.2])**.** *For $q \geq 5mp$ and $\alpha \leq 1/(p\sqrt{m} \cdot \omega(\log n))$, the above scheme is correct.*

Proof. Notice that

$$\boldsymbol{d} = \boldsymbol{c} - \boldsymbol{S}^T\boldsymbol{u} = \boldsymbol{P}^T\boldsymbol{e} + \boldsymbol{w} - \boldsymbol{S}^T\boldsymbol{A}\boldsymbol{e} = \boldsymbol{X}^T\boldsymbol{e} + \boldsymbol{w}.$$

Let $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_l]$. Then, to prove the correctness, we need to show that for any $i \in [l]$, $|\boldsymbol{x}_i^T\boldsymbol{e}| < q/2p$ with overwhelming probability.

We first fix some $i \in [l]$. By the construction, $\boldsymbol{x} \leftarrow \lfloor q\boldsymbol{y} \rceil \bmod q$, where $\boldsymbol{y} \leftarrow N(0, \alpha^2/2\pi)^m$. Consider $\boldsymbol{x}' \leftarrow \lfloor q\boldsymbol{y} \rceil$. Notice that if $|\boldsymbol{x}'^T\boldsymbol{e}| < q/2p$ then $|\boldsymbol{x}^T\boldsymbol{e}|$ is also smaller than $q/2p$. Hence, it suffices to show that $|\boldsymbol{x}'^T\boldsymbol{e}| < q/2p$ with overwhelming probability.

By the construction, we have that $\|\boldsymbol{x}' - q\boldsymbol{y}\| \leq \sqrt{m}/2$. In addition, we have that

$$|\boldsymbol{x}'^T\boldsymbol{e}| \leq |(\boldsymbol{x}' - q\boldsymbol{y})^T\boldsymbol{e}| + q|\boldsymbol{y}^T\boldsymbol{e}| \leq m/2 + q|\boldsymbol{y}^T\boldsymbol{e}|$$

by the Cauchy–Schwartz bound. Since, $q > 5mp$, it suffices to show $|\boldsymbol{y}^T\boldsymbol{e}| < 2/5p$. (if so, we have $m/2 + q|\boldsymbol{y}^T\boldsymbol{e}| < q/(10p) + 4q/(10p) = q/2p$.)

Since Gaussian has a regenerativity, the random variable $\boldsymbol{y}^T\boldsymbol{e}$ is distributed as the Gaussian whose variance is at most $m\alpha^2/2\pi$. Thus, we have that for any $\boldsymbol{e} \in \{0,1\}^m$,

$$\Pr[|\boldsymbol{y}^T\boldsymbol{e}| > 2/5p] \leq \frac{\sqrt{m}\alpha}{\sqrt{2\pi}2/5p} \cdot \exp\left(-\pi\frac{(2/5p)^2}{m\alpha^2}\right) = \exp(-\omega(\log n)),$$

since $\alpha < 1/(p\sqrt{m} \cdot \omega(\sqrt{\log n}))$. □

Notes: The original Regev encryption scheme is parametrized by $p = 2$ and $l = 1$. Kawachi, Tanaka, and Xagawa [KTX07] improved the ciphertext blowup by factor $O(\log n)$ by set $p = n^c \geq 2$ for some constant $c > 0$ without changing the public key. Peikert, Vaikuntanathan, and Waters [PVW08] proposed amortizing technique $l \geq 1$. Micciancio and Regev [MR08] changed the domain of the randomness $\{0, 1, \dots, \sigma\}^m$ instead of $\{0,1\}^m$.

12.8.2 Security Proof

First, we can change the real key $[\boldsymbol{A}; \boldsymbol{P}^T]$ with the fake key $[\boldsymbol{A}; \boldsymbol{P}^T]$ which is distributed over $U(\mathbb{Z}_q^{(n+l)\times m})$ under the dLWE assumption. If we use the fake key, the ciphertext contains no information of a message, since $[\boldsymbol{A}; \boldsymbol{P}^T]$ is uniformly distributed and it is universal as the hash functions.

Theorem 12.8.4 (IND-CPA Security, adapted [Reg09, KTX07, PVW08])**.** *Let $m \geq ((1+\delta)n + l)\log q$ for some constant $\delta > 0$. The scheme* LWE-PKE *is IND-CPA secure under the* dLWE(q, χ) *assumption.*

Combining this theorem and the arguments in Section 2.4.3, the security is reduced from the quantum hardness of SIVP_γ or GapSVP_γ.

Proof. Let ϵ denote the advantage of the adversary $\mathcal{A}$ against IND-CPA game.

We consider the following $l+1$ games. In Game_i, the public key is computed as follows: First, $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$. Next, take samples $\boldsymbol{s}_{i+1}, \dots, \boldsymbol{s}_l \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{x}_{i+1}, \dots, \boldsymbol{x}_l \leftarrow \chi^m$. Then, compute $\boldsymbol{p}_j \leftarrow \boldsymbol{A}^T \boldsymbol{s}_j + \boldsymbol{x}_j$ for $j = i+1, \dots, l$. In addition, take i samples $\boldsymbol{p}'_1, \dots, \boldsymbol{p}'_i \leftarrow \mathbb{Z}_q^m$. The public key is $\boldsymbol{A}$ and $\boldsymbol{P} = [\boldsymbol{p}'_1, \dots, \boldsymbol{p}'_i, \boldsymbol{p}_{i+1}, \dots, \boldsymbol{p}_l]$.

Game_0 is the original IND-CPA game. In addition, in Game_l, the public key is now uniformly at random. Let S_i be an event that the adversary $\mathcal{A}$ wins in Game_i. Then, we have that

$$|\Pr[S_0] - 1/2| = \epsilon \text{ and } |\Pr[S_l] - 1/2| \le q^{-\frac{1}{2}\delta n} = \mathsf{negl}(n).$$

The latter inequation follows from the leftover hash lemma (see Section 4.3.1). By using the hybrid argument, there is an index $i \in [l]$ such that

$$|\Pr[S_{i-1}] - \Pr[S_i]| \ge \epsilon/l - q^{-\delta n/2}/l.$$

However, if there exists such index, we can solve the $\mathrm{dLWE}_{q,\chi}$ problem as follows: Take the m samples $(\boldsymbol{A}, \boldsymbol{p}^*)$ from the oracle of $\mathrm{dLWE}_{q,\chi}$ problem. Then, make a public key $(\boldsymbol{A}, \boldsymbol{P})$, where $\boldsymbol{p}'_j \leftarrow \mathbb{Z}_q^m$ for $j = 1, \dots, i-1$, $\boldsymbol{p}_i = \boldsymbol{p}^*$, and $\boldsymbol{p}_j = \boldsymbol{A}^T \boldsymbol{s}_j + \boldsymbol{x}_j$ for $j = i+1, \dots, l$. If the oracle is $U(\mathbb{Z}_q^{n+1})$, then the simulation is for Game_{i-1}. If the oracle is $A_{\boldsymbol{s},\chi}$, then the simulation is for Game_i. Hence, we have that

$$\mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n) \ge \epsilon/l - q^{-\delta n/2}/l.$$

This shows that

$$\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{LWE\text{-}PKE},\mathcal{A}}(n) \le l \cdot \mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n) + q^{-\delta n/2}$$

and completes the proof. □

12.8.3 Attacks

The following TB-CCA1 attack is due to Izmerly and Mor [IM06] and Xagawa.

For simplicity, we consider $\mathsf{R05}$ with $p = 2$ and $l = 1$. By these specification, the public parameter is $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, the secret key is $\boldsymbol{s} \in \mathbb{Z}_q^n$, the public key is $\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s}+\boldsymbol{x} \in \mathbb{Z}_q^m$, the encryption of the message $v \in \{0,1\}$ is $(\boldsymbol{u}, c) = (\boldsymbol{A}\boldsymbol{e}, \boldsymbol{p}^T\boldsymbol{e}+v\lfloor q/2\rceil)$ for some $\boldsymbol{e} \in \{0,1\}^m$. The decryption algorithm outputs 0 if $d = c - \boldsymbol{u}^T\boldsymbol{s}$ is close to 0, outputs 1 otherwise. Specifically, the decryption algorithm outputs 0 if $|d| \le q/4$.

We describe how to extract the first coordinate s_1 of the secret key $\boldsymbol{s}$. The other coordinates are extracted by a slight modification. Let t denote $\lfloor q/4\rfloor$. Let us set $\boldsymbol{u} = (1, 0, \dots, 0)$. Then, in decryption, the variable d is set to be $c - s_1 \bmod q$, where s_1 is the first coordinate of the secret key, $\boldsymbol{s}$. Sliding c, we can detect when d is firstly larger than t since the response switches from 0 to 1. Let c^* denote the value such that $c^* - s_1 = t$. By solving this, we have $s_1 = c^* - t$.

12.8.4 Extensions

As mentioned in the introduction, there are several security proofs of the variant of the scheme LWE-PKE. Akavia, Goldwasser, and Vaikuntanathan showed the key-leakage security of the scheme under $\mathrm{dLWE}_{q,\chi}$ assumption in the smaller dimension $n' < n$.

Applebaum, Cash, Peikert, and Sahai [ACPS09] proposed a simple variant of LWE-PKE, where $q = p^2$ and the secret key is chosen from χ^n and $\boldsymbol{s} \in \mathbb{Z}_p^n \subset \mathbb{Z}_q^n$. They showed the key-dependent message security of their variant; in the key-dependent message CPA game, the adversary chooses a function $f_{t,w} : \mathbb{Z}_p^n \to \mathbb{Z}_p$, where $f_{t,w}(\boldsymbol{s}) = \langle \boldsymbol{t}, \boldsymbol{s} \rangle + w \bmod p$.

In addition, Lyubashevsky, Palacio, and Segev [LPS10] proposed that the variant of LWE-PKE whose security is based on the subset sum problem.

12.9 The Gentry–Peikert–Vaikuntanathan "Dual" encryption scheme

They observed that the "dual" of LWE-PKE is also a public-key encryption scheme: The public key is $(\boldsymbol{A}, \boldsymbol{u} = \boldsymbol{A}\boldsymbol{e})$ and the ciphertext is $(\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x}, c = \boldsymbol{u}^T\boldsymbol{s} + x + w\lfloor q/2 \rceil)$. The decryption is done by computing $d \leftarrow v - \boldsymbol{e}^T\boldsymbol{p} = w\lfloor q/2 \rceil - \boldsymbol{e}^T\boldsymbol{x} + x$ and rounding it.

The point is the public key $(\boldsymbol{A}, \boldsymbol{u})$ is uniformly chosen from $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n$. This yields an identity-based encryption by combining the GPV signature scheme in Section 11.3 (see Section 14.3). We note that the distribution of the public key $(\boldsymbol{A}, \boldsymbol{p})$ of LWE-PKE is somewhat sparse in $\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^m$.

12.9.1 Description

Scheme 12.9.1 (Dual [GPV08])**.**

Setup(1^n)**:** On input the security parameter 1^n, output the random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ as *param*.

KeyGen(*param* = A)**:** Generate $\boldsymbol{E} \leftarrow D^l_{\mathbb{Z}^m,s}$. The encryption key is $ek = \boldsymbol{U} \leftarrow \boldsymbol{A}\boldsymbol{E} \in \mathbb{Z}_q^{n\times l}$. The decryption key is $dk = \boldsymbol{E} \in \mathbb{Z}^{m\times l}$.

Enc($ek = \boldsymbol{U}, msg = \boldsymbol{b}$)**:** Generate $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{x}_p \leftarrow \chi^m$, and $\boldsymbol{x}_v \leftarrow \chi^l$. Compute $\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^m$. In addition, compute $\boldsymbol{v} = \boldsymbol{U}^T\boldsymbol{s} + \boldsymbol{x}_v \in \mathbb{Z}_q^l$. For message $\boldsymbol{b} \in \mathbb{Z}_p^l$, compute $\boldsymbol{w} = t(\boldsymbol{b}) \in \mathbb{Z}_q^l$. Then, the ciphertext is $(\boldsymbol{p}, \boldsymbol{c} = \boldsymbol{v} + \boldsymbol{w})$.

Dec($dk = \boldsymbol{e}, ct = (\boldsymbol{p}, \boldsymbol{c})$) . Compute $\boldsymbol{d} = \boldsymbol{c} - \boldsymbol{E}^T\boldsymbol{p} \in \mathbb{Z}_q^l$. Output the plaintext $\boldsymbol{b} \in \mathbb{Z}_p^l$ with $d_i - t(b_i) \in \mathbb{Z}_q$ is closest to 0.

Theorem 12.9.2 (Correctness, [GPV08, Theorem 7.1])**.** *Let* $\chi = \bar{\Psi}_\alpha$*,* $q \geq 5(m+1)ps$*, and* $\alpha \leq 1/(ps\sqrt{m+1} \cdot \omega(\log n))$*. The above scheme is correct.*

Proof. For simplicity, we only show the correctness where $l = 1$. The other case can be proved in the same manner. Notice that

$$d = c - \boldsymbol{e}^T\boldsymbol{p} = \boldsymbol{u}^T\boldsymbol{s} + x_v + w - \boldsymbol{e}^T\boldsymbol{A}^T\boldsymbol{s} - \boldsymbol{e}^T\boldsymbol{x}_p = w + x_v - \boldsymbol{e}^T\boldsymbol{x}_p.$$

Hence, to prove the correctness, we will show that $|\boldsymbol{x}_v - \boldsymbol{e}^T\boldsymbol{x}_p| < q/2p$ over $\mathbb{Z}$. Notice that by the construction, we have $\|\boldsymbol{e}\| \leq s\sqrt{m}$ with overwhelming probability.

To obtain the above upperbound, it suffices to show that $|\boldsymbol{x}^T\boldsymbol{e}| < q/2p$ where $\boldsymbol{x} \leftarrow \bar{\Psi}_\alpha^{m+1}$ and for any $\boldsymbol{e} \in \mathbb{Z}^{m+1}$ such that $\|\boldsymbol{e}\| \leq s\sqrt{m+1}$.

As in the previous correctness proof of Theorem 12.8.3, we replace $\boldsymbol{x}$ with $\boldsymbol{x}'$ and $\boldsymbol{y}$. By the construction, $\boldsymbol{x} \leftarrow \lfloor q\boldsymbol{y} \rceil \bmod q$, where $\boldsymbol{y} \leftarrow N(0, \alpha^2/2\pi)^{m+1}$. Consider $\boldsymbol{x}' \leftarrow \lfloor q\boldsymbol{y} \rceil$. Notice that if $|\boldsymbol{x}'^T\boldsymbol{e}| < q/2p$ then $|\boldsymbol{x}^T\boldsymbol{e}|$ is also smaller than $q/2p$. Hence, it suffices to show that $|\boldsymbol{x}'^T\boldsymbol{e}| < q/2p$ with overwhelming probability. By the construction, we have that $\|\boldsymbol{x}' - q\boldsymbol{y}\| \leq \sqrt{m+1}/2$. In addition, we have that

$$|\boldsymbol{x}'^T\boldsymbol{e}| \leq |(\boldsymbol{x}' - q\boldsymbol{y})^T\boldsymbol{e}| + q|\boldsymbol{y}^T\boldsymbol{e}| \leq s(m+1)/2 + q|\boldsymbol{y}^T\boldsymbol{e}|$$

by the Cauchy–Schwartz bound. Since, $q > 5(m+1)ps$, it suffices to show $|\boldsymbol{y}^T\boldsymbol{e}| < 2/5p$. (if so, we have $s(m+1)/2 + q|\boldsymbol{y}^T\boldsymbol{e}| < q/(10p) + 4q/(10p) = q/2p$.)

Since Gaussian has a regenerativity, the random variable $\boldsymbol{y}^T\boldsymbol{e}$ is distributed as the Gaussian whose variance is $\|\boldsymbol{e}\|\,\alpha^2/2\pi$. Thus, we have that for any $\boldsymbol{e}$ with $\|\boldsymbol{e}\| \leq s\sqrt{m+1}$,

$$\Pr[|\boldsymbol{y}^T\boldsymbol{e}| > 2/5p] = \exp(-\omega(\log n)),$$

since $\alpha < 1/(ps\sqrt{m+1} \cdot \omega(\sqrt{\log n}))$. □

12.9.2 Security Proof

It easy show the IND-CPA security assuming the $\mathrm{dLWE}_{q,\chi}$ is hard on average. Notice that $\boldsymbol{U}$ is distributed almost uniformly over $\mathbb{Z}_q^{n\times l}$ if $s = \omega(\log n)$.

Theorem 12.9.3 (IND-CPA Security, adapted [GPV08]). *Let $m \geq 2(n+l)\log q$ and $s = \omega(\sqrt{\log m})$. Then the scheme* Dual *is IND-CPA secure under the* $\mathrm{dLWE}_{q,\chi}$ *assumption.*

Proof. Assume that there exists an adversary $\mathcal{A}$ that wins the IND-CPA game with advantage ϵ.

We consider the following 3 games. Game_0 is the original IND-CPA game. In Game_1, we replace the public key with $(\boldsymbol{A}, \boldsymbol{U}) \leftarrow \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times l}$. In Game_2, we replace the generation method of the challenge ciphertext as follows: Let $(\boldsymbol{p}', \boldsymbol{v}') \leftarrow U(\mathbb{Z}_q^m \times \mathbb{Z}_q^l)$. Then, the target ciphertext is $(\boldsymbol{p}', \boldsymbol{c}' = \boldsymbol{v}' + \boldsymbol{t})$.

Let S_i be an event that the adversary $\mathcal{A}$ wins in Game_i. Then, we have that

$$|\Pr[S_0] - 1/2| = \epsilon, |\Pr[S_0] - \Pr[S_1]| \leq \mathsf{negl}(n), |\Pr[S_2] - 1/2| = 0.$$

The second inequation follows from Corollary 10.5.5 and our parameter settings. In addition, we have

$$|\Pr[S_1] - \Pr[S_2]| \leq \mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n).$$

Taking m samples $(\boldsymbol{A}, \boldsymbol{p})$ and l samples $(\boldsymbol{U}, \boldsymbol{v})$ from the oracle of the dLWE problem, simulate the game with the adversary $\mathcal{A}$. If the oracle is $A_{s,\chi}$, we have simulated Game_1, otherwise, we have simulated Game_2. Then, it follows.

These argument show that

$$\epsilon = \mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{LWE\text{-}PKE},\mathcal{A}}(n) \leq \mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n) + \mathsf{negl}(n)$$

and completes the proof. □

12.9.3 Attacks

The TB-CCA1 attack below follows the attack by Izmerly and Mor [IM06] and Xagawa.

For simplicity, we consider Dual with $p = 2$ and $l = 1$. By these specification, the public parameter is $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, the secret key is $\boldsymbol{e} \in \mathbb{Z}^m$, the public key is $\boldsymbol{u} = \boldsymbol{A}\boldsymbol{e} \in \mathbb{Z}_q^n$, the encryption of the message $b \in \{0, 1\}$ is $(\boldsymbol{p}, c) = (\boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x}_p, \boldsymbol{u}^T\boldsymbol{s} + x_v + b\lfloor q/2\rceil)$. The decryption algorithm outputs 0 if $d = c - \boldsymbol{e}^T\boldsymbol{p}$ is close to 0, outputs 1 otherwise. Specifically, the decryption algorithm outputs 0 if $|d| \leq q/4$.

The idea is same as that in the attack against $\mathsf{R05}$. We describe how to extract the first coordinate e_1 of the secret key $\boldsymbol{e}$. The other coordinates are extracted by a slight modification. Let t denote $\lfloor q/4\rfloor$. Let us set $\boldsymbol{p} = (1, 0, \ldots, 0)$. Then, in decryption, the variable d is set to be $c - e_1 \bmod q$. Sliding c, we can detect when d is firstly larger than t since the response switches from 0 to 1. Let c^* denote the value such that $c^* - e_1 = t$. By solving this, we have $e_1 = c^* - t$.

12.10 The Peikert–Waters “Lossy” Trapdoor Functions

Peikert and Waters [PW08] defined lossy trapdoor functions (LTDFs) and all-but-one trapdoor functions (ABO TDFs).

Intuitively, LTDFs have two mode: lossy mode and invertible mode. In invertible mode, the legitimate user with a trapdoor can invert the his function. However, in lossy mode, any user cannot information theoretically. In addition, the keys in lossy mode and invertible mode are computationally indistinguishable.

The precise definition is given as follows: Let n be the security parameter, and $\lambda(n)$ represent the input length of the function, $\kappa(n)$ represent the lossiness of the collection. For convenience, we define the residual leakage $\rho(n) = \lambda(n) - \kappa(n)$.

Definition 12.10.1 (Lossy trapdoor functions, [PW08])**.** Consider a following scheme $\mathsf{LosTDF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$. Let us suppose that $\mathsf{mode} \in \{\mathsf{inj}, \mathsf{los}\}$.

$\mathsf{Gen}(1^n, \mathsf{mode})$**:** A generation algorithm, given the security parameter 1^n, and works as follows:

- If $\mathsf{mode} = \mathsf{inj}$, it outputs (a, t).
- If $\mathsf{mode} = \mathsf{los}$, it outputs $(a, \perp)$.

These defines the functions $f_a : \{0, 1\}^\lambda \to R_n$.

$\mathsf{Eval}(a, x)$**:** An evaluation algorithm, given the index a and $x \in \{0, 1\}^{\lambda}$, it outputs $f_a(x)$.

$\mathsf{Inv}(a, t, y)$**:** An inversion algorithm, given the set of (a, t) generate by $\mathsf{Gen}(1^n, \mathsf{inj})$ and $y = f_a(x) \in R_n$, it outputs x.

We say LosTDF is a collection of (λ, κ)-lossy trapdoor functions if the following conditions hold:

Easy to sample an injective function with trapdoor: Consider $(a, t) \leftarrow \mathsf{Gen}(1^n, \mathsf{inj})$. Then, the function f_a is an *injective* function and Inv with input $(a, t, y = f_a(x))$ efficiently retrieves x.

Easy to sample a lossy function: Consider $(a, \perp) \leftarrow \mathsf{Gen}(1^n, \mathsf{los})$. Then, $f_a : \{0, 1\}^{\lambda} \to R_n$ has image size at most $2^{\rho} = 2^{\lambda-\kappa}$, that is, $\left|f_a(\{0, 1\}^{\lambda})\right| \leq 2^{\rho}$.

Hard to distinguish injective from lossy: The first outputs of $\mathsf{Gen}(1^n, \mathsf{inj})$ and $\mathsf{Gen}(1^n, \mathsf{los})$ are computationally indistinguishable.

Definition 12.10.2 (All-but-one trapdoor functions, [PW08])**.** Consider a following scheme $\mathsf{ABOTDF} = (\mathsf{Gen}, \mathsf{Eval}, \mathsf{Inv})$. Let V_n be a set of branches.

$\mathsf{Gen}(1^n, v^*)$**:** A generation algorithm, given the security parameter 1^n and lossy branch $v^* \in V_n$, and outputs (a, t), where a is a function index and t is its trapdoor. The index a defines the function $g_a(\cdot, \cdot) : V_n \times \{0, 1\}^{\lambda} \to R_n$.

$\mathsf{Eval}(a, v, x)$**:** An evaluation algorithm, given the index a, a branch $v \in V_n$, and $x \in \{0, 1\}^{\lambda}$, it outputs $g_a(v, x) = g_{a,v}(x)$.

$\mathsf{Inv}(t, v, y)$**:** An inversion algorithm, given the trapdoor t and $y = g_{a,v}(x) \in R_n$, if $v \neq v^*$ outputs x.

We say ABOTDF is a collection of (λ, κ)-all-but-one trapdoor functions if the following conditions hold:

Easy to sample an injective function with trapdoor: Consider $(a, t) \leftarrow \mathsf{Gen}(1^n, v^*)$. Then, for any $v \neq v^*$, the function $g_{a,v}$ is an *injective* function and Inv with input $(t, v, y = g_{a,v}(x))$ efficiently retrieves x.

Loss on the lossy branch: Consider $(a, t) \leftarrow \mathsf{Gen}(1^n, v^*)$. Then, $g_{a,v^*} : \{0, 1\}^{\lambda} \to R_n$ has image size at most $2^{\rho} = 2^{\lambda-\kappa}$, that is, $\left|g_{a,v^*}(\{0, 1\}^{\lambda})\right| \leq 2^{\rho}$.

Hidden lossy branch: Consider the following game: An adversary $\mathcal{A}$ outputs $(v_0, v_1) \in V_n^2$, is given a function index a, where $(a, t) \leftarrow \mathsf{Gen}(1^n, v_b)$ and $b \leftarrow \{0, 1\}$, and outputs b'. For any polynomial-time adversary $\mathcal{A}$,

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \leq \mathsf{negl}(n).$$

It is easy to show that LTDFs are one-way. In addition, LTDFs yields ABOTDfs with two branches and the l-time use of $(n, n - r)$-ABOTDFs with branch set $V = \{0, 1\}$ yields $(n, n - lr)$-ABOTDfs with branch set $V = \{0, 1\}^l$. Furthermore, they yields pseudorandom generators, collision-resistant hash families, and thus one-time secure signature schemes. (See [PW08, Section 3]). Using them, they construct encryption schemes and oblivious transfers.

Peikert and Waters instantiated LTDFs and ABO TDFs based on the DDH assumption and the LWE assumption. Following their result, many researchers proposed LTDFs and ABO TDFs under several assumptions.

It is worth to note how we obtain the idea of lossy trapdoor functions. Recall LWE-PKE. The ciphertext of $\boldsymbol{b}$ with a randomness $\boldsymbol{e}$ is

$$\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{c}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{P}^T\end{bmatrix}\cdot\boldsymbol{e} + \begin{bmatrix}\boldsymbol{0}\\ \lfloor q/p\rfloor\,\boldsymbol{b}\end{bmatrix}$$

and a legitimate receiver can retrieve the vector $\boldsymbol{b}$ rather than $\boldsymbol{e}$. To retrieve $\boldsymbol{e}$, we set $l = m$ and consider the following function

$$\begin{bmatrix}\boldsymbol{u}\\ \boldsymbol{c}\end{bmatrix} = \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{P}^T + \lfloor q/p\rfloor\,\boldsymbol{I}_m\end{bmatrix}\cdot\boldsymbol{e} = \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{P}^T\end{bmatrix}\cdot\boldsymbol{e} + \begin{bmatrix}\boldsymbol{0}\\ \lfloor q/p\rfloor\,\boldsymbol{e}\end{bmatrix}.$$

Obviously, we can retrieve $\boldsymbol{e}$. This is the main idea of Peikert and Waters and the DDH construction is done by the same idea. However, to apply this idea to LWE-PKE, they need to circumvent several obstacles, for example, the noise may leak the information and we cannot ensure lossiness. We describe the circumventions in the following sections.

12.10.1 Descriptions of Lattice-Based Lossy Functions

Matrix Concealer

We start with recalling matrix concealer, which makes the function index. This definition is as known as *matrix encryption*.

$\mathsf{GenConceal}_\chi(1^n, m, l)$**:** The inputs are the security parameter 1^n, and integers $m, l = \mathrm{poly}(n)$. First generate two random matrices $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{S} \leftarrow \mathbb{Z}_q^{n\times l}$ and an error matrix $\boldsymbol{X} \leftarrow \chi^{m\times l}$. Then output $\boldsymbol{C} = [\boldsymbol{A}; \boldsymbol{P}^T]$, where $\boldsymbol{P} = \boldsymbol{A}^T\boldsymbol{S} + \boldsymbol{X} \in \mathbb{Z}_q^{m\times l}$.

Although notation is changed, this algorithm is the same as the key-generation algorithm of LWE-PKE. Hence, the output $\boldsymbol{C}$ of $\mathsf{GenConceal}_\chi$ is computationally indistinguishable from $U(\mathbb{Z}_q^{(n+l)\times m})$ if $m, l = \mathrm{poly}(n)$ and $\mathrm{dLWE}_{q,\chi}$ is hard on average. The proof is obtained by the hybrid argument on the columns of $\boldsymbol{P}$.

The following lemma will be used later.

Lemma 12.10.3 ([PW08])**.** *Let h, w, p be positive integers. Let $q \geq 4ph$, let $1/\alpha \geq 8p(m+g)$ for some $g > 0$, and let $\chi = \bar{\Psi}_\alpha$. Then except with probability at most $w\cdot 2^{-g}$ over the choice of $\boldsymbol{X} \leftarrow \chi^{m\times l}$, the following holds: for every row vector $\boldsymbol{e} = (e_1, \ldots, e_m) \in \{0,1\}^m$, each entry of $\frac{1}{q}\boldsymbol{X}\boldsymbol{e} \in \mathbb{T}^l$ has absolute value less than $\frac{1}{4p}$.*

Lossy TDFs

For a while we assume that $p = 2^k$ for some $k \geq 1$. Define a special row vector $\boldsymbol{g} = [1, 2, \ldots, 2^{\log p - 1} = p/2] \in \mathbb{Z}^k$. Then, we define $\boldsymbol{G} = \boldsymbol{I}_h \otimes \boldsymbol{g} \in \mathbb{Z}_q^{h \times hk}$, where $\otimes$ denotes the Tensor product. Illustratively, we have

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{g} & \boldsymbol{0} & \ldots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{g} & \ldots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \ldots & \boldsymbol{g} \end{bmatrix}$$

By using 2-base representation, we can define the invertible function $\mathsf{encode}(w) = \boldsymbol{e} = (e_1, \ldots, e_k) \in \{0, 1\}^k$ such that $\boldsymbol{g} \cdot \boldsymbol{e} = w$ for any $w \in \mathbb{Z}_p$. We map $\boldsymbol{w} = (w_1, \ldots, w_l) \in \mathbb{Z}_p^h$ into $\boldsymbol{e} = \mathsf{encode}(\boldsymbol{w}) \in \{0, 1\}^{hk}$ and vice verse. Then we have that $\boldsymbol{Ge} = \boldsymbol{w}$.

The description of the construction is given as follows:

Scheme 12.10.4 ([PW08])**.** Let us set $m = lk$.

$\mathsf{Gen}(1^n, \mathsf{mode} \in \{\mathsf{inj}, \mathsf{los}\})$**:** The algorithm first invokes $\mathsf{GenConceal}_\chi(1^n, m, l)$ to generate a matrix $\boldsymbol{C} = [\boldsymbol{A}; \boldsymbol{P}^T] \in \mathbb{Z}_q^{(n+l) \times m}$ and a trapdoor $\boldsymbol{S} \in \mathbb{Z}_q^{n \times l}$, where $\boldsymbol{P} = \boldsymbol{A}^T \boldsymbol{S} + \boldsymbol{X}$.

- If $\mathsf{mode} = \mathsf{inj}$, output the function index $\boldsymbol{Y} = [\boldsymbol{A}; \boldsymbol{P}^T + \boldsymbol{M}] \in \mathbb{Z}_q^{(n+l) \times m}$ and the trapdoor $\boldsymbol{S}$, where $\boldsymbol{M} = t(\boldsymbol{G})$.
- If $\mathsf{mode} = \mathsf{los}$, output the function index $\boldsymbol{Y} = \boldsymbol{C}$.

$\mathsf{Eval}(\boldsymbol{Y}, \boldsymbol{e})$**:** Let $\boldsymbol{Y}$ be a function index and $\boldsymbol{e} \in \{0, 1\}^m$. Output $\boldsymbol{z} = \boldsymbol{Ye} \in \mathbb{Z}_q^{(n+l)}$.

$\mathsf{Inv}(\boldsymbol{S}, z)$**:** Parse z as $(\boldsymbol{u}, \boldsymbol{v}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$. Then compute $\boldsymbol{d} \leftarrow \boldsymbol{v} - \boldsymbol{S}^T \boldsymbol{u}$ and let $\boldsymbol{w} = t^{-1}(\boldsymbol{d}) \in \mathbb{Z}_p^l$. Finally output $\boldsymbol{e} \leftarrow \mathsf{decode}(\boldsymbol{w}) \in \{0, 1\}^m$.

The index-generation algorithm is the same as LWE-PKE.KeyGen if $\mathsf{mode} = \mathsf{los}$. But, in the case where $\mathsf{mode} = \mathsf{inj}$, the key is changed to $\boldsymbol{P} + \boldsymbol{M}$. Crucially, this change allows us to recover $\boldsymbol{e}$ with the trapdoor $\boldsymbol{S}$. It is obvious that the adversary distinguishes $[\boldsymbol{A}; \boldsymbol{P}^T + \boldsymbol{M}]$ from $[\boldsymbol{A}; \boldsymbol{P}^T]$ yields the adversary distinguishes $[\boldsymbol{A}; \boldsymbol{P}^T]$ and $[\boldsymbol{A}; \boldsymbol{P}'^T]$, where $\boldsymbol{P}'$ is drawn from $\mathbb{Z}_q^{m \times l}$ uniformly at random.

The correctness follows from the correctness conditions of LWE-PKE. The main part is the following lossiness proof.

Theorem 12.10.5 ([PW08])**.** *Let $q \geq 4lp \log p$ and $\chi = \bar{\Psi}_\alpha$ with $1/\alpha \geq 16pm = 16lp \log p$. Then the above algorithms define a collection of almost-always (m, m')-lossy TDFs under the* $\mathrm{dLWE}_{q,\chi}$ *assumption, where $m = n \log p$ and the residual leakage $r = m - m'$ is*

$$r \leq \frac{m}{l} \left(n + (n + l) \log_p (q/p) \right).$$

Proof. Lossiness is computed as follows: Let $\boldsymbol{Y} = [\boldsymbol{A} | \boldsymbol{P}^T = \boldsymbol{S}^T \boldsymbol{A} + \boldsymbol{X}^T]$ be a function index produced by $\mathsf{Gen}(1^n, \mathsf{los})$.

$$\mathsf{Eval}(\boldsymbol{Y}, \boldsymbol{e}) = (\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{Ye} = (\boldsymbol{Ae}, \boldsymbol{S}^T \boldsymbol{Ae} + \boldsymbol{X}^T \boldsymbol{e}).$$

The number of possible values for $\boldsymbol{u}$ is at most q^n. Given $\boldsymbol{u}$, the number of possible values for $\boldsymbol{v}$is exactly the number of possible values for $\boldsymbol{X}^T\boldsymbol{e}$. The latter quantity is at most $(1+q/2p)^l \le (q/p)^l$. Hence the total number of outputs is at most $q^n \cdot (q/p)^l$. Therefore,

$$\begin{aligned} r &\le n \cdot \log q + l \cdot \log(q/p) = m \cdot \frac{n \log q}{l \log p} + \frac{m}{\log p} \cdot \log(q/p) \\ &= \frac{m}{l}\left(n + (n+l)\log_p(q/p)\right). \end{aligned}$$

□

We omit how to construct all-but-one trapdoor functions from $\mathrm{dLWE}_{q,\chi}$ assumption but note they construct directly all-but-one trapdoor functions from as in the above rather than general construction we already mentioned. For the details, see [PW08].

12.10.2 Description of Encryption Scheme

After the constructions of lossy TDFs and all-but-one TDFs, they gave an IND-CCA2 secure encryption scheme based on them. We review the construction here.

Scheme 12.10.6 (PW-PKE [PW08]). Let $\mathsf{LosTDF} = (\mathsf{L.Gen}, \mathsf{L.Eval}, \mathsf{L.Inv})$ and $\mathsf{ABOTDF} = (\mathsf{A.Gen}, \mathsf{A.Eval}, \mathsf{A.Inv})$ be (λ, κ)-lossy and (λ, κ')-ABO trapdoor functions with branch set $V = \{0,1\}^v$. We require the total residual leakage is

$$\rho + \rho' = 2\lambda - \kappa - \kappa' \le \lambda - a,$$

for some $a = a(n) = \omega(\log n)$. Let $\mathsf{OTS} = (\mathsf{O.KeyGen}, \mathsf{O.Sign}, \mathsf{O.Ver})$ be a one-time secure signature scheme with verification-key space $V = \{0,1\}^v$. Let $\mathcal{H}$ be a universal family of hash functions from $\{0,1\}^\lambda \to \{0,1\}^l$, where $0 < l \le a - 2\log 1/\epsilon$ for some negligible $\epsilon = \epsilon(n)$. The message space is $\{0,1\}^l$.

$\mathsf{KeyGen}(1^n)$**:** Given 1^n, it generates $(a,t) \leftarrow \mathsf{L.Gen}(1^n, \mathsf{inj})$, $(a',t') \leftarrow \mathsf{A.Gen}(1^n, 0^v)$, and a hash function $h \leftarrow \mathcal{H}$. The encryption key is $ek = (a, a', h)$ and the decryption key is $dk = (t, t', ek)$.

$\mathsf{Enc}(ek = (a,a',h), msg \in \{0,1\}^l)$**:** It generates a key pair $(vk, sk) \leftarrow \mathsf{O.KeyGen}(1^n)$, chooses $x \leftarrow \{0,1\}^\lambda$ uniformly at random. It computes

$$c_1 = \mathsf{L.Eval}(a,x) = f_a(x), c_2 = \mathsf{A.Eval}(a', vk, x) = g_{a',vk}(x), c_3 = m \oplus h(x).$$

Finally, it signs the triplet (c_1,c_2,c_3) as $\sigma \leftarrow \mathsf{O.Sign}(sk, (c_1,c_2,c_3))$. Then, it outputs the ciphertext $ct = (vk, c_1, c_2, c_3, \sigma)$.

$\mathsf{Dec}(dk = (t,t',ek), ct = (vk, c_1, c_2, c_3, \sigma)$**:** It first checks that $\mathsf{O.Ver}(vk, (c_1,c_2,c_3), \sigma) = 1$. If not, it outputs $\perp$ and halts. It then computes $x \leftarrow \mathsf{L.Inv}(t, c_1)$, and checks that $c_1 = f_a(x)$ and $c_2 = g_{a',vk}(x)$; if not, it outputs $\perp$ and halts. Finally, it outputs $m \leftarrow c_3 \oplus h(x)$.

Theorem 12.10.7 ([PW08, Theorem 4.2])**.** *The above scheme is IND-CCA2 secure.*

We only give the games in the proof and intuitions.

- The game Game_0 is the original IND-CCA2 game, which induces the target ciphertext $ct^* = (vk^*, c_1^*, c_2^*, c_3^*, \sigma^*)$.
- In Game_1, they changed the decryption oracle which rejects the query $ct = (vk, c_1, c_2, c_3, \sigma)$ if $vk = vk^*$. If it happens, the one-time security of OTS is violated.
- In Game_2, they replace the lossy branch 0^v with vk^*. The distance between Game_1 and Game_2 is ensured by the hidden branch property.
- In Game_3, the decryption oracles retrieve x by $\mathsf{A.Inv}(t', vk, c_2)$ instead of $\mathsf{L.Inv}(t', c_1)$. This change makes no difference since f_a and $g_{a,vk}$ is injective.
- In Game_4, they replace the injective function with a lossy function. The distance between Game_1 and Game_2 is ensured by the “hard to distinguish injective from lossy” property.
- In Game_5, they replace the component c_3^* with a uniformly random string over $\{0,1\}^l$. Since h is extractor, this only makes a statistical difference.

We note that we have no need to sign c_1. This observation is due to Matsuda, Nishimaki, and Tanaka [MNT10], who proposed an IND-CCA2 secure proxy re-encryption scheme based on the LTDFs based on the DDH assumption.

Notes: Unfortunately, the above scheme instantiated from the dLWE assumption has very huge public key, say $\tilde{O}(n^3)$, and thus it is not convenient to use the scheme in the real world. However, the power of lossiness is curious and attractive, and the proof techniques are very useful and powerful. We require the more efficient construction of an IND-CCA2 secure encryption scheme.

13
Key-Encapsulation Mechanism

Organization: Section 13.1 and Section 13.2 reviews the definitions on key-encapsulation mechanism (KEM) and data-encapsulation mechanism (DEM), respectively. Section 13.3 reminds us of the construction of PKE from KEM and DEM, that is, the KEM/DEM framework. Section 13.4 gives the description of Peikert's KEM. We give a review of ideal-lattice-based versions of the encryption schemes by Stehlé, Steinfeld, Tanaka, and Xagawa in Section 13.5.

13.1 Definitions of Key-Encapsulation Mechanism

13.1.1 Model of Key-Encapsulation Mechanism

A key-encapsulation mechanism scheme KEM with associated key space K_n is a triplet of algorithms $(\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$.

$\mathsf{Gen}(1^n)$**:** A key-generation algorithm, given 1^n, outputs a pair of an encryption key and a decryption key (ek, dk).

$\mathsf{Encaps}(ek, msg)$**:** An encapsulation algorithm, given ek, outputs a key $k \in K_n$ and a ciphertext ct.

$\mathsf{Decaps}(dk, ct)$**:** A decapsulation algorithm, given dk and ct, returns a key k or a special symbol $\perp$.

Correctness: The correctness of a key-encapsulation mechanism is defined as follows: With overwhelming probability the ciphertext of any key $k \in K_n$ under an encryption key ek should be decrypted into k, where the probability is taken by coins of Gen and Encaps. Formally, this requirement is denoted

$$\Pr\left[k \neq \tilde{k} : \begin{array}{l} (ek, dk) \leftarrow \mathsf{Gen}(1^n); \\ (k, ct) \leftarrow \mathsf{Encaps}(ek); \\ \tilde{k} \leftarrow \mathsf{Decaps}(dk, ct); \end{array}\right] \leq \mathsf{negl}(n).$$

13.1.2 Security Notions

We adopt the standard security notions [HHK06], indistinguishability of ciphertexts under several attacks. Roughly speaking, we say the scheme has indistinguishability if any polynomial-time adversary cannot distinguish a valid key k_1 from a random key k_0 with the ciphertext ct of a valid key. In chosen plaintext attacks (cpa), the adversary could only encrypt its chosen message and cannot use the decryption oracle. In chosen ciphertext attacks (cca1), the adversary could query to the decryption oracle until the adversary commits the target messages. In chosen ciphertext attacks (cca2), the adversary could query to the decryption oracle after it receives the target ciphertext.

We describe the formal definition as follows: Consider the experiment $\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$, where atk $\in$ $\{\text{cpa}, \text{cca1}, \text{cca2}\}$.

Experiment $\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(n)$:

Setup Phase: The challenger takes the security parameter n and obtains $param \leftarrow \mathsf{Setup}(1^n)$ and $(ek, dk) \leftarrow \mathsf{Gen}(param)$. It gives $param$ and ek to the adversary $\mathcal{A}$.

Learning Phase 1: The adversary can issue queries to the oracle if atk $\in$ $\{\text{cca1}, \text{cca2}\}$. The oracle Dec receives an input ct and returns $k \leftarrow \mathsf{Decaps}(dk, ct)$.

Challenge Phase: The adversary $\mathcal{A}$ query to the challenger. The challenger generates a random key $k_0 \leftarrow K_n$ and a pair of a valid key and ciphertext $(k_1, ct) \leftarrow \mathsf{Encaps}(param, ek)$. The challenger flips a coin $b \leftarrow \{0, 1\}$, sets $k^* \leftarrow k_b$, and sends (k^*, ct^*) to the adversary.

Learning Phase 2: The adversary can issue queries to the oracle if atk $=$ cca2. The oracle Dec receives input ct. If $ct = ct^*$, the challenger outputs 0 and halts. Otherwise, the oracle returns $k \leftarrow \mathsf{Decaps}(dk, ct)$ to $\mathcal{A}$.

Guessing Phase: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$, the challenger outputs 1, otherwise 0.

Definition 13.1.1. Let $\mathsf{KEM} = (\mathsf{Setup}, \mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be a key-encapsulation mechanism, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(n) = \left|\Pr\left[\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say that KEM is ind-atk secure if $\mathbf{Adv}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$, where atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$.

13.2 Definitions of Data-Encapsulation Mechanism

13.2.1 Model of Data-Encapsulation Mechanism

A data-encapsulation mechanism scheme DEM with associated key space K_n and message space M_n is a triplet of algorithms ($\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps}$).

$\mathsf{Gen}(1^n)$**:** A key-generation algorithm, given the security parameter 1^n, outputs a key $k \in K_n$.

$\mathsf{Encaps}(k, msg)$**:** An encapsulation algorithm, given a key k and a data $msg \in M_n$, outputs a ciphertext ct.

$\mathsf{Decaps}(k, ct)$**:** A decapsulation algorithm, given k and ct, returns a message msg or a special symbol $\perp$.

Correctness: The correctness of a data-encapsulation mechanism is defined as follows: With overwhelming probability the ciphertext of any message msg in the message space under an encryption key k should be decrypted into msg, where the probability is taken by coins of Gen, and Encaps. Formally, this requirement is denoted

$$\Pr\left[msg \neq \widetilde{msg} : \begin{array}{c} k \leftarrow \mathsf{Gen}(1^n); \\ ct \leftarrow \mathsf{Encaps}(k, msg); \\ \widetilde{msg} \leftarrow \mathsf{Decaps}(k, ct); \end{array} \right] \leq \mathsf{negl}(n).$$

13.2.2 Security Notions

We adopt the standard security notions [BDJR97, HHK06], indistinguishability of ciphertexts under several attacks. Roughly speaking, we say the scheme has indistinguishability if any polynomial-time adversary cannot distinguish a valid key k_1 from a random key k_0 with the ciphertext ct of a valid key.

In chosen plaintext attacks (cpa), the adversary could only encrypt its chosen message and cannot use the decryption oracle. In chosen ciphertext attacks (cca1), the adversary could query to the decryption oracle until the adversary commits the target messages. In chosen ciphertext attacks (cca2), the adversary could query to the decryption oracle after it receives the target ciphertext. In these attacks the adversary could query to the *encryption* oracle.

There are other attacks. In addition, Herranz, Hofheinz, and Kiltz formalized one-time security for DEM, which was already appeared in the other names in the literature [CS03, KY06]. In one-time attacks (ot), the adversary has no oracles. In one-time chosen-ciphertext attacks (otcca), the adversary could query to the decryption oracle after obtaining the challenge ciphertext. In there attacks, the adversary has no encryption oracle.

We describe the formal definition as follows: Consider the experiment $\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$, where atk $\in$ $\{\text{ot}, \text{otcca}, \text{cpa}, \text{cca1}, \text{cca2}\}$.

Experiment $\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{PKE},\mathcal{A}}(n)$:

Setup Phase: The challenger takes the security parameter n and obtains $param \leftarrow \mathsf{Setup}(1^n)$ and $(ek, dk) \leftarrow \mathsf{KeyGen}(param)$. It gives $param$ and ek to the adversary $\mathcal{A}$.

Learning Phase 1: The adversary can issue queries to the encryption oracle if atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$. In addition, it can query to the decryption oracle if atk $\in \{\text{cca1}, \text{cca2}\}$.

- The oracle Enc receives an input msg and returns $ct \leftarrow \mathsf{Encaps}(k, msg)$.
- The oracle Dec receives an input ct and returns $msg \leftarrow \mathsf{Decaps}(k, ct)$.

Challenge Phase: The adversary $\mathcal{A}$ query two distinct message $msg_0, msg_1 \in M_n$ to the challenger. The challenger flips a coin $b \leftarrow \{0, 1\}$ and sends $ct^* \leftarrow \mathsf{Encaps}(k, msg_b)$ to the adversary.

Learning Phase 2: The adversary can issue queries to the encryption oracle if atk $\in \{\text{cpa}, \text{cca1}, \text{cca2}\}$. It also can query to the decryption oracle if atk $\in \{\text{otcca}, \text{cca2}\}$.

- The oracle Enc receives an input msg and returns $ct \leftarrow \mathsf{Encaps}(k, msg)$.
- The oracle Dec receives an input ct. If $ct = ct^*$, the challenger outputs 0 and halts. Otherwise, the oracle returns $msg \leftarrow \mathsf{Decaps}(k, ct)$ to $\mathcal{A}$.

Guessing Phase: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$, the challenger outputs 1, otherwise 0.

Definition 13.2.1. Let $\mathsf{DEM} = (\mathsf{Gen}, \mathsf{Encaps}, \mathsf{Decaps})$ be a data-encapsulation mechanism, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ind-atk}}_{\mathsf{DEM},\mathcal{A}}(n) = \left|\Pr\left[\mathbf{Exp}^{\text{ind-atk}}_{\mathsf{DEM},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say that DEM is ind-atk secure if $\mathbf{Adv}^{\text{ind-atk}}_{\mathsf{KEM},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$, where atk $\in \{\text{ot}, \text{otcca}, \text{cpa}, \text{cca1}, \text{cca2}\}$.

13.3 Hybrid Encryption

We here briefly review the framework of the hybrid encryption, the construction of a public-key encryption scheme from a key- and data-encapsulation mechanism.

Scheme 13.3.1 (Hybrid Encryption)**.** Let $\mathsf{KEM} = (\mathsf{K.Gen}, \mathsf{K.Encaps}, \mathsf{K.Decaps})$

be a KEM and let $\mathsf{DEM} = (\mathsf{D.Gen}, \mathsf{D.Encaps}, \mathsf{D.Decaps})$ be a DEM. Then, the obtained scheme $\mathsf{PKE} = (\mathsf{P.KeyGen}, \mathsf{P.Enc}, \mathsf{P.Dec})$ is defined as follows:

$\mathsf{P.KeyGen}(1^n)$**:** Output $(ek, dk) \leftarrow \mathsf{K.Gen}(1^n)$.

$\mathsf{P.Enc}(ek, msg)$**:** Obtain $(k, ct_1) \leftarrow \mathsf{K.Encaps}(ek)$, obtain $ct_2 \leftarrow \mathsf{D.Encaps}(k, msg)$, and outputs $ct = (ct_1, ct_2)$.

$\mathsf{P.Dec}(dk, ct = (ct_1, ct_2))$**:** Retrieve $k \leftarrow \mathsf{K.Decaps}(dk, ct_1)$ and outputs $msg \leftarrow \mathsf{D.Decaps}(k, ct_2)$.

Herranz et al. showed the following results. The final statement is already proved by Cramer and Shoup [CS03].

- For any atk $\in \{\text{ot}, \text{otcca}, \text{cpa}, \text{cca1}, \text{cca2}\}$, if KEM is ind-cpa secure and DEM is ind-atk secure, then the obtained PKE is ind-cpa secure.
- For any atk $\in \{\text{ot}, \text{otcca}, \text{cpa}, \text{cca1}, \text{cca2}\}$, if KEM is ind-cca1 secure and DEM is ind-atk secure, then the obtained PKE is ind-cca1 secure.
- For any atk $\in \{\text{ot}, \text{cpa}, \text{cca1}\}$, if KEM is ind-cca2 secure and DEM is ind-atk secure, then the obtained PKE is ind-cca1 secure.
- For any atk $\in \{\text{otcca}, \text{cca2}\}$, if KEM is ind-cca2 secure and DEM is ind-atk secure, then the obtained PKE is ind-cca2 secure.

13.4 Peikert's Key-Encapsulation Mechanism and Public-Key Encryption Schemes

We have already seen that the key generation and encryption in the McEliece encryption scheme resemble to the key-generation method of the Regev encryption scheme LWE-PKE. Why cannot we use the lattice-based analogy of the McEliece encryption scheme? Can we replace $\boldsymbol{G}' \in \mathbb{F}^{n \times m}$ with $\boldsymbol{A} \in \mathbb{Z}_q^{n \times m}$ and the distribution $U(\mathcal{S}(m, t))$ with χ^m?

Gentry, Peikert, and Vaikuntanathan [GPV08] pointed out that the short basis $\boldsymbol{T}$ of the lattice $\Lambda_q^{\perp}(\boldsymbol{A})$ also exploits the lattice $\Lambda_q(\boldsymbol{A})$. Using $\boldsymbol{T}$, we can solve the BDD over the lattice $\Lambda_q(\boldsymbol{A})$; that is, given $\boldsymbol{p} = \boldsymbol{A}^T \boldsymbol{s} + \boldsymbol{x}$, we can recover $\boldsymbol{s}$ and $\boldsymbol{x}$! (We will describe the simplified variant later.) One-wayness of this function, with respect to the input distribution $U(\mathbb{Z}_q^n) \times \chi^m$, is apparent under the sLWE assumption.

We can interpret this results into the analogy of the McEliece encryption scheme as follows: Instead of the Hamming weight, we consider the Lee weight (see Roth's textbook [Rot06]). For an element $c \in \mathbb{Z}_q$, we define the *Lee value* $|c|$ by

$$|c| = \begin{cases} c, & 0 \le c \le q/2, \\ q - c & q/2 < c \le q - 1 \end{cases}$$

as we already defined the absolute value $|c|$ for $c \in \mathbb{Z}_q$. The *Lee weight* of a vector $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{Z}_q^n$ is defined by $\sum_{i=1}^n |c_j|$. This is just the l_1 norm of $\boldsymbol{c}$ if $\boldsymbol{c}$

is embedded in $[-q/2, q/2]^n$. The recovering algorithm in Gentry et al. shows, a trapdoor $\boldsymbol{T}$ of $\boldsymbol{A}$ enables us to *decode* a received word $\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x} \bmod q$ into $\boldsymbol{s}$ if the *Lee weight* of $\boldsymbol{x}$ is small.

13.4.1 Basic Schemes

Peikert [Pei09c] improved the invert method of the LWE trapdoor function mentioned in [GPV08]. Now, the obtained scheme has a flavor of the GGH encryption scheme in Section 12.5.

Scheme 13.4.1 (LWETrap [Pei09c])**.**

Setup(1^n)**:** Given the security parameter 1^n, output 1^n.

KeyGen(1^n)**:** Using ExtLattice, obtain $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$, where $\|\boldsymbol{T}\| \le L$.

Eval($ek = \boldsymbol{A}, msg = \boldsymbol{s}$)**:** Let $\boldsymbol{s} \in \mathbb{Z}_q^n$. Choose $\boldsymbol{x} \leftarrow \Psi_\alpha^m$ and compute $\boldsymbol{p} \leftarrow \frac{1}{q}\boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x} \in \mathbb{T}^m$. Output $g_{\boldsymbol{A}}(\boldsymbol{s},\boldsymbol{x}) = \bar{\boldsymbol{p}} = \lfloor q' \cdot \boldsymbol{p} \rceil \bmod q'$.

Inv($dk = \boldsymbol{T}, \bar{\boldsymbol{p}}$)**:** Given $\bar{\boldsymbol{p}} = g_{\boldsymbol{A}}(\boldsymbol{s},\boldsymbol{x}) \in \mathbb{Z}_{q'}^m$, let $\boldsymbol{p}' \leftarrow \bar{\boldsymbol{p}}/q \in \mathbb{T}^m$, compute $\boldsymbol{y} \leftarrow \boldsymbol{T}^{-T} \cdot \left\lfloor \boldsymbol{T}^T\boldsymbol{p}' \right\rceil \bmod 1$, and computes $\boldsymbol{s}'$ from $\boldsymbol{y}$ by solving $\boldsymbol{y} = \frac{1}{q}\boldsymbol{A}^T\boldsymbol{s} \in \mathbb{T}^m$. (It can also output $\boldsymbol{x}' \leftarrow \boldsymbol{p}' - \frac{1}{q}\boldsymbol{A}^T\boldsymbol{s} \in \mathbb{T}^m$.)

The correctness of decryption follows from the similar argument in the proof of Lemma 12.5.1. For appropriately chosen α, we can show that the norm of $\boldsymbol{x}$ is short.

Theorem 13.4.2 (Correctness, [Pei09c, Lemma 4.2])**.** *Let $q' = q'(n) \ge 2L\sqrt{m}$ and $1/\alpha \ge L \cdot \omega(\sqrt{\log n})$. Then for any $\boldsymbol{s} \in \mathbb{Z}_q^n$ and for $\boldsymbol{x}$ drawn from Ψ_α^m, the inversion algorithm on input $\bar{\boldsymbol{p}} = g_{\boldsymbol{A}}(\boldsymbol{s},\boldsymbol{x})$ correctly outputs $\boldsymbol{s}$ with overwhelming probability over the choice of $\boldsymbol{x}$.*

Using this trapdoor function, we can construct key-encapsulation mechanism with key space $\{0,1\}^l$ which resembles Dual in Section 12.9.

Scheme 13.4.3 (LWE-KEM, combinded, [Pei09c] and [Pei09b])**.** Let LWETrap = (L.KeyGen, L.Eval, L.Inv) as in the above.

Gen(1^n)**:** Generate $(\boldsymbol{A},\boldsymbol{T}) \leftarrow$ L.KeyGen. Generate $\boldsymbol{U} \leftarrow \mathbb{Z}_q^{n\times l}$. It outputs $ek = (\boldsymbol{A},\boldsymbol{U})$ and $dk = (\boldsymbol{T}, ek)$.

Encaps(ek)**:** Choose a key $\boldsymbol{k} \leftarrow \{0,1\}^l$. Generate a random vector $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and generate $\boldsymbol{x}_p \leftarrow \chi^m$ and $\boldsymbol{x}_v \leftarrow \chi^l$. Then compute $\bar{\boldsymbol{p}} \leftarrow g_{\boldsymbol{A}}(\boldsymbol{s},\boldsymbol{x}_p) =$ L.Eval$(\boldsymbol{A},\boldsymbol{s};\boldsymbol{x}_p)$ and $\bar{\boldsymbol{v}} \leftarrow g_{\boldsymbol{U}}(\boldsymbol{s},\boldsymbol{x}_v)$. Compute $\bar{\boldsymbol{c}} \leftarrow \bar{\boldsymbol{v}} + \lfloor q'/2 \rfloor \boldsymbol{k} \bmod q'$. Finally, output $\boldsymbol{k}$ and $ct = (\bar{\boldsymbol{p}}, \bar{\boldsymbol{c}})$.

Decaps($dk, ct = (\bar{\boldsymbol{p}}, \bar{\boldsymbol{c}})$)**:** Retrieve $\boldsymbol{s} \leftarrow$ L.Inv$(dk, \bar{\boldsymbol{p}})$. Compute $\boldsymbol{v} \leftarrow \boldsymbol{U}^T\boldsymbol{s}/q \bmod 1$. Then, compute $\boldsymbol{d} \leftarrow \bar{\boldsymbol{c}} - \lfloor q'\boldsymbol{v} \rceil \bmod q'$. Finally, output $\boldsymbol{k} \leftarrow t^{-1}(\boldsymbol{d})$.

Theorem 13.4.4. *The above KEM is* ind-cpa *secure if* dLWE(q,χ) *is hard. More* *precisely, for any polynomial-time adversary $\mathcal{A}$,*

$$\mathbf{Adv}^{\text{ind-cpa}}_{\mathsf{LWE\text{-}KEM},\mathcal{A}}(n) \leq \mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n) + \mathsf{negl}(n).$$

Proof. Consider three games;

Game_0**:** The original game.

Game_1**:** In the game, we change the generation method for $\boldsymbol{A}$, $\boldsymbol{U}$, $\bar{\boldsymbol{p}}$, and $\bar{\boldsymbol{v}}$; Take $m + l$ samples from $A_{\boldsymbol{s},\chi}$. Let us name the first m samples $(\boldsymbol{A}, \boldsymbol{p})$ and $(\boldsymbol{U}, \boldsymbol{v})$. Compute $\bar{\boldsymbol{p}} = \lfloor q'\boldsymbol{p}\rceil \bmod q'$ and $\bar{\boldsymbol{v}} \leftarrow \lfloor q'\boldsymbol{v}\rceil \bmod q'$ and use them in the game.

Game_2**:** In the game, we change the generation method for $\boldsymbol{A}$, $\boldsymbol{U}$, $\bar{\boldsymbol{p}}$, and $\bar{\boldsymbol{v}}$; Take $m + l$ samples from $U(\mathbb{Z}_q^n \times \mathbb{T})$. Let us name the first m samples $(\boldsymbol{A}, \boldsymbol{p})$ and $(\boldsymbol{U}, \boldsymbol{v})$. Compute $\bar{\boldsymbol{p}} = \lfloor q'\boldsymbol{p}\rceil \bmod q'$ and $\bar{\boldsymbol{v}} \leftarrow \lfloor q'\boldsymbol{v}\rceil \bmod q'$ and use them in the game.

Apparently, the statistical distance between Game_0 and Game_1 is at most $\mathsf{negl}(n)$ since they only differ the generation method of $\boldsymbol{A}$. In addition, the distance between Game_1 and Game_2 is at most $\mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n)$; otherwise, $\mathcal{A}$ distinguishes $A_{\boldsymbol{s},\chi}$ and $U(\mathbb{Z}_q^n \times \mathbb{T})$. This completes the proof. □

13.4.2 CCA Schemes

Using the LWE trapdoor function, Goldwasser and Vaikuntanathan [GV08] and Peikert [Pei09c] constructed IND-CCA2 secure encryption schemes by employing the Rosen–Segev construction [RS09]. (We note that Dowsley, Müller-Quade, and Nascimento [DMQN09] also constructed an IND-CCA2 secure encryption scheme from the McEliece encryption scheme based on the assumption on the coding problems.)

The Rosen–Segev construction is summarized as follows:

Scheme 13.4.5 (The Rosen–Segev Construction [RS09])**.** Let $\mathsf{Trap} = (\mathsf{T.Gen}, \mathsf{T.Eval}, \mathsf{T.Inv})$ be one-way functions. Let $\mathsf{OTS} = (\mathsf{O.Gen}, \mathsf{O.Sign}, \mathsf{O.Ver})$ be a one-time secure signature scheme with verification-key space is $\{0,1\}^v$. Assume that the trapdoor function $f_a : \{0,1\}^n \to \{0,1\}^n$ is one-way. We also assume that $f_{\boldsymbol{a}}(s) = (f_{a_1}(s), \ldots, f_{a_k}(s))$ is also one-way. Additionally, a function $h : \{0,1\}^n \to \{0,1\}^l$ is a hardcore function of $f_{\boldsymbol{a}}$.

$\mathsf{P.KeyGen}(1^n)$**:** For $i \in [v]$ and $b \in \{0,1\}$, obtain $(a_i^{(b)}, t_i^{(b)}) \leftarrow \mathsf{T.Gen}$. Output $ek = \{a_i^{(b)}\}$ and $dk = (\{t_i^{(b)}\}, ek)$.

$\mathsf{P.Enc}(ek, msg)$**:** Generate $(vk, sk) \leftarrow \mathsf{O.Gen}(1^n)$. Generate a random string $s \leftarrow \{0,1\}^n$. Compute $c_1 \leftarrow (f_{a_1^{(vk_1)}}(s), \ldots, f_{a_v^{(vk_v)}}(s))$ and $c_2 \leftarrow h(s) \oplus msg$. Obtain a signature $\sigma \leftarrow \mathsf{O.Sign}(sk, (c_1, c_2))$. Output the ciphertext $ct = (vk, c_1, c_2, \sigma)$.

$\mathsf{P.Dec}(dk, ct)$**:** Verify the signature σ; output $\perp$ if $\mathsf{O.Ver}(vk, (c_1, c_2)) = 0$. Then, by using $\mathsf{T.Inv}$, invert $f_{a_1^{(vk_1)}}(s)$ and obtain s. Confirm the other $f_{a_i^{(vk_i)}}(s)$ by $\mathsf{T.Eval}$; output $\perp$ if not. Then, obtain a message $msg = h(s) \oplus c_2$.

Showing that the scheme is IND-CCA2 secure is educationally simple as Rosen and Segev noted [RS09]. The intuition is that the simulator implants given $f_{\boldsymbol{a}} = (f_{a_1}, \dots, f_{a_k})$ into $\{f_{a_i^{(vk_i)}}\}$ for its chosen vk^*, and simulates the decryption oracle if $vk^* \neq vk$.

Returning to the lattice-based scheme, the function $g_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{x})$ is secure with respect to the distribution $U(\mathbb{Z}_q^n) \times \chi^{m \times k}$; It is obvious that the new function $g_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{x}) = (g_{\boldsymbol{A}_1}(\boldsymbol{s}, \boldsymbol{x}_1), \dots, g_{\boldsymbol{A}_k}(\boldsymbol{s}, \boldsymbol{x}_k))$ is also one-way under the sLWE assumption, where $\boldsymbol{A} = [\boldsymbol{A}_1 | \dots | \boldsymbol{A}_k]$, $\boldsymbol{x} = \boldsymbol{x}_1 \circ \dots \circ \boldsymbol{x}_k$, $\boldsymbol{s}$ is chosen from $\mathbb{Z}_q^n$ uniformly at random, each $\boldsymbol{x}_i$ is a sample from χ^m. It is also true that the above function $g_{\boldsymbol{A}}$ is pseudorandom under the dLWE assumption.

There are some difficulties for direct applying the Rosen–Segev technique. Since we cannot recover $\boldsymbol{x}$ (we recover $\boldsymbol{x}'$ in the above trapdoor function), this noise will be exploited by the IND-CCA2 adversary. In addition, the simulator in the IND-CCA2 game have to able to check some $\boldsymbol{p}'$ is correctly generated under $f_{\boldsymbol{A}}$ even if it does not know its trapdoor. To circumvent this difficulties, Peikert defines the preimage verification algorithm for $g_{\boldsymbol{A}}$.

PreVer$(A, (s, x'), \bar{\boldsymbol{p}})$**:** Compute $\boldsymbol{p}' \leftarrow \bar{\boldsymbol{p}}/q' \in \mathbb{T}^m$. Accept if $\|\boldsymbol{x}\|_\infty < \alpha \cdot t$ and $\boldsymbol{b}' = \frac{1}{q}\boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x}' \in \mathbb{T}^m$ and reject otherwise.

Lemma 13.4.6 ([Pei09c, Lemma 4.4])**.** *For $q' \geq 1/(\alpha t) \geq 2L\sqrt{m} \geq 8$, the algorithm* PreVer *and* LWETrap *satisfies the following conditions:*

Completeness: *For any $\boldsymbol{s}$ and $\boldsymbol{x}$ drawn from Ψ_α^m, and $\boldsymbol{x}'$ output by the inversion algorithm given $\bar{\boldsymbol{p}} = g_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{x})$ and $\boldsymbol{T}$,* PreVer$(\boldsymbol{A}, (\boldsymbol{s}, \boldsymbol{x}'), \bar{\boldsymbol{p}})$ *accepts with overwhelming probability over the choice of $\boldsymbol{x}$.*

Unique preimage: *For every $\bar{\boldsymbol{p}} \in \mathbb{Z}_{q'}^m$, there is at most one legal preimage $(\boldsymbol{s}, \boldsymbol{x}')$ under $g_{\boldsymbol{A}}$; that is,* PreVer$(\boldsymbol{A}, (\boldsymbol{s}, \boldsymbol{x}'), \bar{\boldsymbol{p}})$ *accepts for at most one value of $(\boldsymbol{s}, \boldsymbol{x}')$.*

Findable preimage: *For any $\bar{\boldsymbol{p}}$, the inversion algorithm, given inputs $\bar{\boldsymbol{p}}$ and $\boldsymbol{T}$, always outputs the unique legal preimage $(\boldsymbol{s}, \boldsymbol{x}')$, i.e., the $(\boldsymbol{s}, \boldsymbol{x}')$ that makes* PreVer *accept, if such pair exists.*

Assuming the sLWE$_{q,\Psi_\alpha}$ is hard, we can show that $(\boldsymbol{A}, g_{\boldsymbol{A}}(\boldsymbol{s}, \boldsymbol{x})) \sim_c (\boldsymbol{A}, \boldsymbol{p}^*)$, where $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{x} \leftarrow \Psi_\alpha^m$, and $\boldsymbol{p}^* \leftarrow \mathbb{Z}_{q'}^m$.

We here construct KEM rather than public-key encryption. This eliminates use of the hardcore functions. The following scheme is the obtained KEM applying the Rosen–Segev construction.

Scheme 13.4.7 (Pei-KEM, combinded, [Pei09c] and [Pei09b])**.** Let OTS = (O.Gen, O.Sign, O.Ver) be a strongly one-time secure signature scheme with verification-key space $\{0,1\}^v$. Let LWETrap = (L.Gen, L.Eval, L.Inv) as in the above.

Gen(1^n)**:** For $i = 1, \dots, v$ and $b \in \{0,1\}$, generate $2v$ key pairs $(\boldsymbol{A}_i^{(b)}, \boldsymbol{T}_i^{(b)}) \leftarrow$ L.Gen(1^n) such that $\|\tilde{\boldsymbol{T}}_i^{(b)}\| \leq L$. Generate $\boldsymbol{U} \leftarrow \mathbb{Z}_q^{n \times l}$. It outputs $ek =$

$(\{\boldsymbol{A}_i^{(b)}\}_{i,b}, \boldsymbol{U})$ and $dk = (\{\boldsymbol{T}_i^{(b)}\}_{i,b}, ek)$.

Encaps(ek)**:** Choose a key $\boldsymbol{k} \leftarrow \{0,1\}^l$. Next, generate a key pair $(vk, sk) \leftarrow$ O.Gen(1^n). Generate a random vector $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and generate $\boldsymbol{x}_{p,i} \leftarrow \chi^m$ for $i = 1, \ldots, v$ and $\boldsymbol{x}_v \leftarrow \chi^l$. Then compute $\bar{\boldsymbol{p}}_i \leftarrow g_{\boldsymbol{A}_i^{(vk_i)}}(\boldsymbol{s}, \boldsymbol{x}_{p,i})$ and $\bar{\boldsymbol{v}} \leftarrow g_{\boldsymbol{U}}(\boldsymbol{s}, \boldsymbol{x}_v)$. Compute $\bar{\boldsymbol{c}} \leftarrow \bar{\boldsymbol{v}} + \lfloor q'/2 \rfloor \boldsymbol{k} \bmod q'$. Sign $\bar{\boldsymbol{p}}_i$ and $\bar{\boldsymbol{c}}$ as $\sigma \leftarrow$ O.Sign$(sk, (\{\bar{\boldsymbol{p}}_i\}, \bar{\boldsymbol{c}}))$. Finally, output $ct = (vk, \{\bar{\boldsymbol{p}}_i\}, \bar{\boldsymbol{c}}, \sigma)$.

Decaps(dk, ct)**:** Check if O.Ver$(vk, (\{\bar{\boldsymbol{p}}_i\}, \bar{\boldsymbol{c}}), \sigma) = 1$; if not, output $\perp$ and halt. Next, invert $\boldsymbol{s}$ from $\bar{\boldsymbol{p}}_1$. Compute $\boldsymbol{x}_i'$ from $\boldsymbol{s}$ and $\bar{\boldsymbol{p}}_i$. Check if PreVer$(\boldsymbol{A}_i^{(vk_i)}, (\boldsymbol{s}, \boldsymbol{x}_i'), \bar{\boldsymbol{p}}_i) = 1$ for any i; if not, output $\perp$ and halt. Finally, retrieve $\boldsymbol{k}$ from $\bar{\boldsymbol{c}}$ and output $\boldsymbol{k}$.

Combining this KEM and some ind-otcca DEM, we obtain ind-cca2 secure public-key encryption scheme.

Theorem 13.4.8 ([Pei09c])**.** *The above* Pei-KEM *is* ind-cca2 *secure if* OTS *is strongly one-time secure and* dLWE(q, χ) *is hard.*

The proof is obtained combining the arguments of Peikert, and Rosen and Segev. We give only the proof sketch here.

Consider the following games;

Game$_0$**:** The original game. In the challenge phase, the challenger works as follows: $\boldsymbol{k}_0, \boldsymbol{k}_1 \leftarrow \{0,1\}^l$, $(vk^*, sk^*) \leftarrow$ O.Gen(1^n), $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$, $\boldsymbol{x}_{p,i} \leftarrow \chi^m$, $\boldsymbol{x}_v \leftarrow \chi^l$,

$$\begin{aligned}
\boldsymbol{p}_i^* &\leftarrow \frac{1}{q}\boldsymbol{A}_i^{(vk_i^*)}\boldsymbol{s} + \boldsymbol{x}_{p,i}, & \bar{\boldsymbol{p}}_i^* &\leftarrow \lfloor q'\boldsymbol{p}_i^* \rceil \bmod q', \\
\boldsymbol{v}^* &\leftarrow \frac{1}{q}\boldsymbol{U}\boldsymbol{s} + \boldsymbol{x}_v, & \bar{\boldsymbol{v}}^* &\leftarrow \lfloor q'\boldsymbol{v}^* \rceil \bmod q', \\
\bar{\boldsymbol{c}}^* &\leftarrow \bar{\boldsymbol{v}} + \lfloor q'/2 \rfloor \boldsymbol{k} \bmod q', & \sigma &\leftarrow \text{O.Sign}(sk^*, \{\bar{\boldsymbol{p}}_i^*\}, \bar{\boldsymbol{c}}^*).
\end{aligned}$$

Game$_1$**:** We change the timing of the generation of vk^*; A priori to the game, the challenger obtains $(vk^*, sk^*) \leftarrow$ O.Gen(1^n).

Game$_2$**:** We change the specification of the decryption oracle: the decryption oracle returns $\perp$ on the query $ct = (vk^*, c_1, c_2, \sigma)$.

Game$_3$**:** We modify the decryption oracle: On the query $(vk, \{\bar{\boldsymbol{p}}_i\}, \bar{\boldsymbol{c}}, \sigma)$, if $vk = vk^*$ then returns $\perp$ as in the previous game. If not, it scans vk and vk^* and finds an index $j \in [t]$ such that $vk_j \neq vk_j^*$. Then, invert $\bar{\boldsymbol{p}}_j$ and obtain $\boldsymbol{s}$. The other procedure is same to the original. Here, the decryption procedure has no need to use $\boldsymbol{T}_i^{(vk_i^*)}$.

Game$_4$**:** We change the key-generation method. For $i \in [v]$, $\boldsymbol{A}_i^{(vk_i^*)} \leftarrow \mathbb{Z}_q^{n \times m}$.

Game$_5$**:** We change the key generation and the generation method for the target ciphertext. The challenger takes $mv + l$ samples from $A_{s,\chi}$. Then, name them

$(\boldsymbol{A}_i^{(vk_i^*)}, \boldsymbol{p}_i^*)$ and $(\boldsymbol{U}, \boldsymbol{v}^*)$.

Game_6**:** We again change the generations. The challenger takes $mv + l$ samples form $U(\mathbb{Z}_q^n \times \mathbb{T})$.

You can show the distances between each games are negligible. It is obvious that Game_0 and Game_1 are identical since we only change the timing. It is also easy to verify that Game_1 and Game_2 are computationally indistinguishable if OTS is strongly one-time secure. The use of PreVer immediately ensures the statistical indistinguishability between Game_2 and Game_3. In addition, the distance between Game_3 and Game_4 is negligible following from the statistical correctness of L.Gen. Game_4 and Game_5 are computationally indistinguishable because dLWE(q,χ) is hard. Hence, we have the following inequality,

$$2 \cdot \mathbf{Adv}^{\text{ind-cca2}}_{\mathsf{Pei\text{-}KEM}}(n) \le \mathbf{Adv}^{\text{ot}}_{\mathsf{OTS}}(n) + \mathbf{Adv}_{\mathrm{dLWE}(q,\chi)}(n) + \mathsf{negl}(n).$$

13.5 The Stehlé–Steinfeld–Tanaka–Xagawa PKE

Stehlé, Steinfeld, Tanaka, and Xagawa [SSTX09] proposed an ideal-lattice version of Peikert's scheme. It is very natural to consider the replacement $\boldsymbol{A}$ with $\check{\boldsymbol{a}}$ in LWETrap yields a secure one-way trapdoor function. But this replacement induces several difficulties. Before discussions, we describe the ILWETrap.

Scheme 13.5.1 (ILWETrap [SSTX09])**.**

Setup(1^n)**:** Given the security parameter 1^n, output 1^n.

KeyGen(1^n)**:** Using ILPSF.TrapGen, obtain $\check{\boldsymbol{a}} \in R^m_{\mathbf{f},q}$ and $\boldsymbol{T} \in R^{m\times m}_{\mathbf{f}}$ of a short basis of $M^{\perp}(\check{\boldsymbol{a}})$. In the following, $\boldsymbol{T}'$ denote $\mathrm{Rot}_{\mathbf{f}}(\boldsymbol{T})$. We suppose that $\|\boldsymbol{T}'\| \le L$.

Eval$(ek = \check{\boldsymbol{a}}, msg = \boldsymbol{s})$**:** Suppose $\boldsymbol{s} \in \mathbb{Z}_q^n$. Choose $\boldsymbol{x} \leftarrow \Psi_\alpha^{mn}$ and compute $\boldsymbol{p} \leftarrow \frac{1}{q}\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})^T\boldsymbol{s} + \boldsymbol{x} \in \mathbb{T}^{mn}$. Output $g_{\check{\boldsymbol{a}}}(\boldsymbol{s},\boldsymbol{x}) = \bar{\boldsymbol{p}} = \lfloor q' \cdot \boldsymbol{p} \rceil \bmod q'$.

Inv$(dk = \boldsymbol{T}, \bar{\boldsymbol{p}})$**:** Given $\bar{\boldsymbol{p}} = g_{\boldsymbol{A}}(\boldsymbol{s},\boldsymbol{x}) \in \mathbb{Z}^m_{q'}$, let $\boldsymbol{p}' \leftarrow \bar{\boldsymbol{p}}/q \in \mathbb{T}^m$, compute $\boldsymbol{y} \leftarrow \boldsymbol{T}'^{-T} \cdot \left\lfloor \boldsymbol{T}'^T \boldsymbol{p}' \right\rceil \bmod 1$, and computes $\boldsymbol{s}'$ from $\boldsymbol{y}$ by solving $\boldsymbol{y} = \frac{1}{q}\mathrm{Rot}(\check{\boldsymbol{a}})^T\boldsymbol{s} \in \mathbb{T}^m$. (It can also output $\boldsymbol{x}' \leftarrow \boldsymbol{p}' - \frac{1}{q}\mathrm{Rot}_{\mathbf{f}}(\check{\boldsymbol{a}})^T\boldsymbol{s} \in \mathbb{T}^m$.)

The problems are twofold. The one is an efficiency issue and the other is that the function is not pseudorandom if we assume the $\mathbf{f}$-sLWE assumption.

Recovering the efficiency: The transpose operation generally kills the efficiency advantage of the ideal-lattice version, since $\mathrm{Rot}_{\mathbf{f}}(\mathbf{a})^T$ may be not suited for the computation. Hence, we need $\tilde{O}(n^2)$ steps to multiply $\mathrm{Rot}_{\mathbf{f}}(\mathbf{a})^T$ and $\boldsymbol{s}$.

Here, we set $\mathbf{f} = x^n + 1$ and recall the reciprocal polynomial in Section 10.7;

$$\mathrm{rec}(\mathbf{a}) = \mathbf{a}(1/x) \text{ in } R_{\mathbf{f}}.$$

Then, we have that

$$\mathrm{Rot}_{\mathbf{f}}(\mathbf{a})^T \cdot \boldsymbol{s} = \mathrm{Rot}_{\mathbf{f}}(\mathrm{rec}(\mathbf{a})) \cdot \boldsymbol{s} = \mathrm{rec}(\mathbf{a}) \otimes \mathbf{s}.$$

The operation rec takes only a small cost and we can compute $\mathrm{Rot}_{\mathbf{f}}(\mathbf{a})^T \boldsymbol{s}$ with $\tilde{O}(n)$ steps as in the before.

On pseudorandomness: Since $\mathrm{Rot}_{\mathbf{f}}(\mathbf{a})^T$ is very structured, we cannot show the pseudorandomness of $(\mathbf{a}, \mathrm{rec}(\mathbf{a}) \otimes \mathbf{s} + \mathbf{x})$, where $\mathbf{s} \leftarrow R_{\mathbf{f},q}$ and $\mathbf{x} \leftarrow \bar{\Psi}_\alpha'^m$ opposite to the success of the reduction from dLWE to sLWE. To circumvent this, Stehlé et al. [SSTX09] used the hardcore function extract the pseudorandomness. They employed the Goldreich–Levin hardcore functions with Toeplitz matrices [GL89, AC02, HMS04, KY06, KX09] which extract constant bits. In their paper, they assumed that the super-polynomial hardness of sLWE and extract $l = o(n)$ bits.

We left the two open problems; The one is efficient hardcore functions with tighter reductions to $\mathbf{f}$-sLWE problem. The other is showing $\mathbf{f}$-dLWE is hard on the average from the lattice assumptions.

14 Identity-Based Encryption

Organization: We give the brief introduction in Section 14.1. Section 14.2 gives the definitions of schemes and security notions. Section 14.3 reviews the Gentry–Peikert–Vaikuntanathan identity-based encryption. In Section 14.4 we review the construction of hierarchical identity-based encryption schemes.

14.1 Introduction

After proposal of the concept of identity-based cryptosystems by Shamir [Sha85], many researchers have made the efforts on construction of identity-based encryption (IBE) schemes.

This was long-standing open problem in cryptography until 2001. The concrete IBE schemes are constructed by Sakai, Ogishi, and Kasahara [SOK01], Boneh and Franklin [BF03], and Cocks [Coc01]; the first and second ones are based on the pairing assumptions and the last one is based on the quadratic residue assumption.

Roughly speaking, these schemes are obtained by combining the signature schemes and the encryption schemes; Let $H : \{0,1\}^* \to K_n$ be the random oracle, where K_n is an encryption-key space of the underlying encryption scheme. The master makes a key pair $(vk, sk) \leftarrow \mathsf{Sig.KeyGen}(1^n)$ and publishes vk. The user encryption key is $ek_{id} = H(id)$ and the user decryption key is $dk_{id} = \sigma_{id} \leftarrow \mathsf{Sig.Sign}(sk, id)$. Such correspondence yields an IBE scheme in the random oracle model.

Turning our eyes on lattice-based IBEs. The first proposal was done by Gentry, Peikert, and Vaikuntanathan [GPV08], which is obtained by combining $\mathsf{GPV\text{-}FDH}$ with Dual. After their construction, Agrawal and Boyen [AB09], Cash, Hofheinz, and Kiltz [CHK09], and Peikert [Pei09b] proposed IBE schemes secure in the standard models. Cash et al. and Peikert also gave HIBEs on which the techniques are essentially same again. In addition, Cash et al. discussed the use of admissible hash functions to enhance the security to be fully secure and gave a concise proof (see the original paper [CHK09]).

We note that there is another IBE scheme by Boneh and Boyen [BB09], however, we cannot confirm their security.

14.2 Definitions

14.2.1 Model of Identity-based Encryption Schemes

An IBE scheme IBE is a quadruplet of algorithms $(\mathsf{Setup}, \mathsf{Ext}, \mathsf{Enc}, \mathsf{Dec})$.

$\mathsf{Setup}(1^n)$: A setup algorithm, given the security parameter 1^n, outputs public parameters $param$ and a master secret key msk.

$\mathsf{Ext}(msk, id)$: An extraction algorithm, given msk and an identity id, outputs a decryption key of the user dk_{id}.

$\mathsf{Enc}(param, id, msg)$: An encryption algorithm, given $param$, id, and a message msg, outputs a ciphertext ct.

$\mathsf{Dec}(dk_{id}, ct)$: A decryption algorithm, given dk_{id} and ct, returns a message msg.

14.2.2 Model of Hierarchical Identity-based Encryption Schemes

A HIBE scheme HIBE is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Ext}, \mathsf{Delg}, \mathsf{Enc}, \mathsf{Dec})$. $\boldsymbol{id} = (id_1, \dots, id_l)$. $\boldsymbol{id}|_i = (id_1, \dots, id_i)$ the i-th prefix of $\boldsymbol{id}$.

$\mathsf{Setup}(1^n)$: A setup algorithm, given the security parameter 1^n, outputs public parameters $param$ and a master secret key msk.

$\mathsf{Ext}(msk, \boldsymbol{id})$: An extraction algorithm, given msk and an identity $\boldsymbol{id}$ of length at most d, outputs a decryption key of the user dk_{id}.

$\mathsf{Delg}(dk_{\boldsymbol{id}|_{l-1}}, \boldsymbol{id})$: A secret-key delegation algorithm, given a decryption key $dk_{\boldsymbol{id}|_{l-1}}$ for a parent $\boldsymbol{id}|_{l-1}$ and an identity $\boldsymbol{id}$ of length at most d, outputs a decryption key of the user $dk_{\boldsymbol{id}}$ for the user $\boldsymbol{id}$.

$\mathsf{Enc}(param, \boldsymbol{id}, msg)$: An encryption algorithm, given $param$, $\boldsymbol{id}$ of length at most d, and a message msg, outputs a ciphertext ct.

$\mathsf{Dec}(dk_{\boldsymbol{id}}, ct)$: A decryption algorithm, given $dk_{\boldsymbol{id}}$ and ct, returns a message msg.

14.2.3 Security Notions

Roughly speaking, the security denotes the adversary cannot distinguish two ciphertexts of its chosen messages even if it can access to the extraction oracle. We note that there are two modes of attacks. The one is a selective ID mode, where the adversary must commit the target identity at the start of the game. The other is a full ID mode, where the adversary can choose the target identity at the challenge phase. Obviously, if IBE is goal-fID-atk-secure it is also goal-sID-atk-secure.

We start to recall the weaker security notion ind-sID-atk security. Consider the experiment $\mathbf{Exp}^{\text{ind-sID-atk}}_{\mathsf{IBE},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\text{ind-sID-atk}}_{\mathsf{IBE},\mathcal{A}}(n)$:

Initiating Phase: The adversary commits an identity id^* to the challenger, which is the target identity of the adversary.

Setup Phase: The challenger C takes the security parameter 1^n and obtains $(param, msk) \leftarrow \mathsf{Setup}(1^n)$. It gives $param$ to the adversary $\mathcal{A}$.

Learning Phase 1: The adversary can issue queries to the oracle Extract. Additionally, $\mathcal{A}$ can issue queries to the oracle Dec if atk $\in \{\text{cca1}, \text{cca2}\}$.

- The oracle Extract receives input id. If $id = id^*$, the challenger outputs 0 and halts. Otherwise, the oracle responds $dk_{id} \leftarrow \mathsf{Ext}(msk, id)$.
- The oracle Dec receives inputs ct and returns $msg \leftarrow \mathsf{Dec}(dk, ct)$.

Challenge Phase: The adversary $\mathcal{A}$ outputs two plaintexts msg_0 and msg_1. The challenger flips a coin $b \leftarrow \{0, 1\}$, sets the target ciphertext to be $ct^* \leftarrow \mathsf{Enc}(param, id^*, msg_b)$, and sends ct^* to the adversary.

Learning Phase 2: Again, the adversary can issue queries to the oracle Extract. If atk = cca2, it also can issue queries to the oracle Dec.

- The oracle Extract receives input id. If $id = id^*$, the challenger outputs 0 and halts. Otherwise, the oracle responds $dk_{id} \leftarrow \mathsf{Ext}(msk, id)$.
- The oracle Dec receives inputs id and ct. If $id = id^*$ and $ct = ct^*$, the challenger outputs 0 and halts. Otherwise, the oracle returns $msg \leftarrow \mathsf{Dec}(dk_{id}, ct)$.

Guessing Phase: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$ the challenger outputs 1, otherwise 0.

Definition 14.2.1. Let $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be an identity-based encryption scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ind-sID-atk}}_{\mathsf{IBE},\mathcal{A}}(n) = \left|\Pr\left[\mathbf{Exp}^{\text{ind-sID-atk}}_{\mathsf{IBE},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say that IBE is ind-sID-atk secure if $\mathbf{Adv}^{\text{ind-sID-atk}}_{\mathsf{IBE},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

We next define the full ID security. In this mode, the adversary can determine a target ID in the challenge phase.

Consider the experiment $\mathbf{Exp}^{\text{ind-fID-atk}}_{\mathsf{IBE},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Experiment $\mathbf{Exp}^{\text{ind-fID-atk}}_{\mathsf{IBE},\mathcal{A}}(n)$:

Setup Phase: The challenger C takes the security parameter 1^n and obtains $(param, msk) \leftarrow \mathsf{Setup}(1^n)$. It gives $param$ to the adversary $\mathcal{A}$.

Learning Phase 1: The adversary can issue queries to the oracle Extract. Additionally, $\mathcal{A}$ can issue queries to the oracle Dec if atk $\in \{\text{cca1}, \text{cca2}\}$.

- The oracle Extract receives input id. The oracle responds $dk_{id} \leftarrow \mathsf{Ext}(msk, id)$.
- The oracle Dec receives inputs ct and returns $msg \leftarrow \mathsf{Dec}(dk, ct)$.

Challenge Phase: The adversary $\mathcal{A}$ outputs two plaintexts msg_0 and msg_1, and a target identity id^*. If id^* is queried to the oracle Extract in the learning phase 1, the challenger C outputs 0 and halts. Otherwise, the challenger flips a coin $b \leftarrow \{0, 1\}$, sets the target ciphertext to be $ct^* \leftarrow \mathsf{Enc}(param, id^*, msg_b)$, and sends ct^* to the adversary.

Learning Phase 2: Again, the adversary can issue queries to the oracle Extract. If atk = cca2, it can issue queries to the oracle Dec.

- The oracle Extract receives input id. If $id = id^*$, the challenger outputs 0 and halts. Otherwise, the oracle responds $dk_{id} \leftarrow \mathsf{Ext}(msk, id)$.
- The oracle Dec receives inputs id and ct. If $id = id^*$ and $ct = ct^*$, the challenger outputs 0 and halts. Otherwise, the oracle returns $msg \leftarrow \mathsf{Dec}(dk_{id}, ct)$.

Guessing Phase: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$ the challenger outputs 1, otherwise 0.

Definition 14.2.2. Let $\mathsf{IBE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be an identity-based encryption scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ind-fID-atk}}_{\mathsf{IBE},\mathcal{A}}(n) = \left|\Pr\left[\mathbf{Exp}^{\text{ind-fID-atk}}_{\mathsf{IBE},\mathcal{A}}(n) = 1\right] - \frac{1}{2}\right|.$$

We say that IBE is ind-fID-atk secure if $\mathbf{Adv}^{\text{ind-fID-atk}}_{\mathsf{IBE},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

To extend these notation to HIBE, we add the new oracle Delegate. We omit the details of the definitions.

14.3 The Gentry–Peikert–Vaikuntanathan Identity-Based Encryption Scheme

This is the first identity-based encryption scheme based on lattice problems. Intuitively, the public parameter and the master key corresponds the verification key and the secret key of the GPV signature scheme. The decryption key of the identity *id* is the signature σ on the message *id*. Notice that the decryption key of Dual corresponds to σ.

14.3.1 Description

Scheme 14.3.1 (GPV-IBE [GPV08]). We model $H : \{0,1\}^* \to \mathbb{Z}_q^{n\times l}$ as the random oracle.

Setup(n)**:** On input the security parameter n, invoke the trapdoor algorithm LPSF.TrapGen(1^n) in Chapter 10 and obtain $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ and its basis $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ such that $\|\tilde{\boldsymbol{T}}\| \le L$. Output $param = \boldsymbol{A}$ and $msk = \boldsymbol{T}$.

Extract($param = \boldsymbol{A}, id$)**:** On input the identity id, compute $[\boldsymbol{u}_{id,1},\ldots,\boldsymbol{u}_{id,l}] = \boldsymbol{U}_{id} \leftarrow H(id)$. Then, invoke the GPV sampling algorithm $\boldsymbol{e}_{id,i} \leftarrow$ LPSF.SamplePre($\boldsymbol{A}, \boldsymbol{T}, s, \boldsymbol{u}_{id,i}$). Output $dk_{id} = \boldsymbol{E}_{id} = [\boldsymbol{e}_{id,1},\ldots,\boldsymbol{e}_{id,l}]$ as user's decryption key.

Enc($id, msg = \boldsymbol{b}$)**:** First generate $\boldsymbol{U}_{id} \leftarrow H(id)$. Then, generate $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{x} \leftarrow \chi^m$. Compute $\boldsymbol{p} = \boldsymbol{A}^T\boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^m$. For message $\boldsymbol{b} \in \mathbb{Z}_p^l$, compute $\boldsymbol{w} = \mathsf{encode}(\boldsymbol{b}) \in \mathbb{Z}_q^l$. Then, the ciphertext is $(\boldsymbol{p}, \boldsymbol{c} = \boldsymbol{U}^T\boldsymbol{s} + \boldsymbol{w})$.

Dec($dk_{id} = \boldsymbol{E}_{id}, ct = (\boldsymbol{p}, \boldsymbol{c})$) . Compute $\boldsymbol{d} = \boldsymbol{c} - \boldsymbol{E}_{id}^T\boldsymbol{p} \in \mathbb{Z}_q^l$. Output the plaintext $\boldsymbol{b} \in \mathbb{Z}_p^l$ by computing decode($\boldsymbol{d}$).

Theorem 14.3.2 (Correctness, [GPV08])**.** *Let $\chi = \bar{\Psi}_\alpha$, $s \ge L \cdot \omega(\sqrt{\log n})$, $q \ge 5(m+1)ps$ and $1/\alpha \ge ps\sqrt{m+1} \cdot \omega(\sqrt{\log n})$. Then the scheme is correct.*

The proof is obtained by an analogy of one of Theorem 12.9.2.

14.3.2 Security Proof

The security proof is obtained by combination of ones of Theorem 12.9.3 and Theorem 11.3.3. Hence, we omit the proof.

Theorem 14.3.3 (Security, [GPV08])**.** *Let $\chi = \bar{\Psi}_\alpha$, $m \ge 2(n+l)\log q$ and $s = \omega(\sqrt{\log m})$. The above IBE is IND-fID-CPA secure under the* dLWE(q,χ) *assumption.*

14.4 The Cash–Hofheinz–Kiltz Hierarchical Identity-Based Encryption Scheme

Very recently three papers, Agrawal and Boyen [AB09], Cash, Hofheinz, and Kiltz [CHK09], and Peikert [Pei09b], proposed identity-based encryption schemes without the random oracles. Here, we omit description of the Agrawal–Boyen IBE since it is included by the Cash–Hofheinz–Kiltz HIBE in the standard model by setting the depth $d = 1$.

Ideas for identity-based encryption: We first give the idea of the IBEs, which often appears in cryptography. Recall the Peikert KEM. In the scheme, the public key for $vk \in \{0,1\}^l$ is $\boldsymbol{A}_{vk} = [\boldsymbol{A}_1^{(vk_1)}|\dots|\boldsymbol{A}_v^{(vk_v)}]$. We replace vk with $id \in \{0,1\}^v$ and the encryption is done as Dual.

Let $(\{\boldsymbol{A}_i^{(b)}\}, \boldsymbol{u})$ be the public parameter (master's public key). The master has corresponding trapdoors $\{\boldsymbol{T}_i^{(b)}\}$. The extraction is done by as follows: (1) generate $\boldsymbol{A}_{id} = [\boldsymbol{A}_1^{(id_1)}|\dots|\boldsymbol{A}_v^{(id_v)}]$ and generate $\boldsymbol{T}_{id}$ by "extending control" in Section 10.8 and (2) obtain $\boldsymbol{e}_{id} \in \mathbb{Z}^{ml}$ such that $\boldsymbol{A}_{id}\boldsymbol{e}_{id} \equiv \boldsymbol{u} \pmod q$. The user secret key is $\boldsymbol{e}_{id}$. Then, the encryption and decryption procedures are the same to the one of Dual.

To show the security in the standard model, the simulator must extract for any $id \neq id^*$. Thus, the simulator, given id^* from the adversary, implant the challenge into $\boldsymbol{A}_i^{(id_i^*)}$ and generate trapdoors $\boldsymbol{T}_i^{(1-id_i^*)}$. If $id \neq id^*$, there is some index $j \in [\lambda]$ such that $id_j \neq id_j^*$. Hence, using the trapdoor $\boldsymbol{T}_j^{(id_j)}$, the simulator can generate $\boldsymbol{e}_{id}$.

The extraction is simplified by adding $\boldsymbol{A}_0$ into the public parameter. The master generates $(\boldsymbol{A}_0, \boldsymbol{T}_0)$ and generates random matrices $\boldsymbol{A}_i^{(b)}$ for $i \in [\lambda]$ and $b \in \{0,1\}$. Then, the use public key is defined as $\boldsymbol{A}_{id} = [\boldsymbol{A}_0|\boldsymbol{A}_1^{(id_1)}|\dots|\boldsymbol{A}_\lambda^{(id_\lambda)}]$.

To expand the identity space $\{0,1\}^\lambda$ to $\{0,1\}^*$, we can use the collision-resistant hash function $H : \{0,1\}^* \to \{0,1\}^\lambda$. Let $t \leftarrow H(id)$ and redefine $\boldsymbol{A}_{id} = [\boldsymbol{A}_0|\boldsymbol{A}_1^{(t_1)}|\dots|\boldsymbol{A}_\lambda^{(t_\lambda)}]$.

Ideas for hierarchical identity-based encryption: In order to delegate the power of the extraction, we can use "randomized control" in Section 10.8. The maximal depth is set to d. Consider $\boldsymbol{id} = (id_1, \dots, id_k) \in (\{0,1\}^*)^k$ for $k \in [d]$.

The master generates $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times m}$ and $\boldsymbol{C}_{i,j}^{(}b) \in \mathbb{Z}_q^{n\times v}$ for $i \in [d]$, $j \in [\lambda]$, and $b \in \{0,1\}$ and choose $H_i : \{0,1\}^* \to \{0,1\}^\lambda$ for $i \in [d]$.

The user encryption key is defined as follows: For id_i, let us define $\boldsymbol{A}_{i,id_i} = [\boldsymbol{C}_{i,1}^{(t_1)}|\dots|\boldsymbol{C}_{i,\lambda}^{(t_\lambda)}]$, where $(t_1, \dots, t_\lambda) = H_i(id_i)$. For $\boldsymbol{id} = (id_1, \dots, id_k)$, $\boldsymbol{A}_{\boldsymbol{id}} = [\boldsymbol{A}_0|\boldsymbol{A}_{1,id_1}|\dots|\boldsymbol{A}_{k,id_k}]$.

This does not change the spirit of the user encryption key. The split enables us to delegate the basis. For any $\boldsymbol{id} = (id_1, \dots, id_k)$, define $\boldsymbol{id}|k-1 = (id_1, \dots, id_{k-1})$, the parent of $\boldsymbol{id}$. Suppose that the parent $\boldsymbol{id}|k-1$ has a basis $\boldsymbol{T}_{\boldsymbol{id}|k-1}$ with quality $L(k-1)$ of a lattice $\Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}|k-1})$. Cash et al. [CHK09] and Peikert [Pei09b] proposed "delegation of the basis" (or "randomized control") which allows $\boldsymbol{id}|k-1$ to generate a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of a lattice $\Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}})$; Compute a basis $\boldsymbol{T}'$ of the lattice $\Lambda_{\boldsymbol{id}} = \Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}})$ and take samples from $D_{\Lambda_{\boldsymbol{id}},s(k-1)}$, where $s(k-1)$ will be defined later. The obtained basis has a quality $L(k) = s(k-1)\cdot\sqrt{m(k-1)}$, where $m(k-1) = m + (k-1)\lambda v$.

We have introduced the parameters $m(k)$, $L(k)$, $s(k)$ for $k = 0, \dots, d$. They are defined inductively as follows:

$$\begin{aligned} &m_0 = m, && L_0 = L, && s_0 = L\cdot\omega(\sqrt{\log n}),\\ &m_k = m + k\lambda v, && L_k = s_{k-1}\cdot\sqrt{m_{k-1}}\cdot\omega(\sqrt{\log m_{k-1}}), && s_k = L_k\cdot\omega(\sqrt{\log n}). \end{aligned}$$

For simplicity, we let $g(n) = \omega(\sqrt{\log n})$ and obtain

$$\begin{aligned} m_d &= m + d\lambda v, \\ L_d &\leq L \cdot (m_d)^{d/2} \cdot g^d \cdot \omega(\log^{d/2} m_d), \\ s_d &\leq L \cdot (m_d)^{d/2} \cdot g^{d+1} \cdot \omega(\log^{d/2} m_d). \end{aligned}$$

14.4.1 Descriptions

Cash et al. defined $v = m$ and use the Alwen–Peikert constructions 1 and 2 (see Section 10.3.2 and Section 10.3.3).

Scheme 14.4.1 (CHK-HIBE [CHK09])**.** The maximal depth is d. We use $\mathcal{H}_n = \{H : \{0,1\}^* \to \{0,1\}^\lambda\}$ a family of hash functions.

Setup(n)**:** On input the security parameter n, invoke the trapdoor algorithm LPSF.TrapGen(1^n) in Chapter 10 and obtain $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times m}$ and its basis $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ such that $\|\tilde{\boldsymbol{T}}\| \leq L$. Next, generate random matrices $\boldsymbol{C}_{i,j}^{(b)} \leftarrow \mathbb{Z}_q^{n\times m}$ for $i \in [d]$, $j \in [\lambda]$ and $b \in \{0,1\}$. Additionally, choose $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l] \leftarrow \mathbb{Z}_q^{n\times l}$ and $H_i \leftarrow \mathcal{H}_n$. Output $param = (\boldsymbol{A}_0, \boldsymbol{U}, \{\boldsymbol{C}_{i,j}^{(b)}\}, \{H_i\})$ and $msk = \boldsymbol{T}$.

Ext$(param = \boldsymbol{A}_0, msk = \boldsymbol{T}, \boldsymbol{id})$**:** For an identity $\boldsymbol{id} = (id_1, \ldots, id_k)$, define $\boldsymbol{A}_{\boldsymbol{id}} = [\boldsymbol{A}_0|\boldsymbol{A}_{1,id_1}|\ldots|\boldsymbol{A}_{k,id_k}] \in \mathbb{Z}_q^{n\times(\lambda k+1)m}$, where $\boldsymbol{A}_{i,id_i} \leftarrow [\boldsymbol{C}_{i,1}^{(t_1)}|\ldots|\boldsymbol{C}_{i,\lambda}^{(t_\lambda)}] \in \mathbb{Z}_q^{n\times\lambda m}$ for $(t_1, \ldots, t_\lambda) \leftarrow H_i(id_i) \in \{0,1\}^\lambda$. Using a short basis $\boldsymbol{T}$ of $\Lambda_q^\perp(\boldsymbol{A}_0)$, it samples a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of $\Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}})$ and $\boldsymbol{E}_{\boldsymbol{id}} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_l]$, where $\boldsymbol{e}_i \leftarrow$ LPSF.SamplePre$(\boldsymbol{A}_{\boldsymbol{id}}, \boldsymbol{T}_{\boldsymbol{id}}, s(k), \boldsymbol{u}_i)$. Output $dk_{\boldsymbol{id}} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{E}_{\boldsymbol{id}})$.

Delg$(param = A_0, usk_{\boldsymbol{id}|k-1} = (\boldsymbol{T}_{\boldsymbol{id}|k-1}, \boldsymbol{E}_{\boldsymbol{id}|k-1}), \boldsymbol{id})$**:** It will output $usk_{\boldsymbol{id}} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{e}_{\boldsymbol{id}})$. Define $\boldsymbol{A}_{\boldsymbol{id}}$ as in the above. Using a short basis $\boldsymbol{T}_{\boldsymbol{id}|k-1}$ of $\Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}|k-1})$, it construct a short basis $\boldsymbol{T}'$ of the lattice $\Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}})$. Then, it samples a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of $\Lambda_q^\perp(\boldsymbol{A}_{\boldsymbol{id}})$ and $\boldsymbol{E}_{\boldsymbol{id}} = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_l]$ where $\boldsymbol{e}_i \leftarrow$ SamplePre$(\boldsymbol{A}_{\boldsymbol{id}}, \boldsymbol{T}', s(k), \boldsymbol{u}_i)$. Output $dk_{\boldsymbol{id}} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{E}_{\boldsymbol{id}})$.

Enc$(param, id, msg = \boldsymbol{b} \in \mathbb{Z}_p^l)$**:** First generate $\boldsymbol{A}_{\boldsymbol{id}}$ as in the above. Then, generate $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{x} \leftarrow \chi^{km}$. Compute $\boldsymbol{p} = \boldsymbol{A}_{\boldsymbol{id}}^T\boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^{km}$. Compute $\boldsymbol{c} = \boldsymbol{U}_{\boldsymbol{id}}^T\boldsymbol{s} + \boldsymbol{x}' +$ encode$(\boldsymbol{b})$, where $\boldsymbol{x}' \leftarrow \chi^l$. Then, the ciphertext is $(\boldsymbol{p}, \boldsymbol{c})$.

Dec$(usk_{id} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{E}_{\boldsymbol{id}}), ct = (\boldsymbol{p}, \boldsymbol{c}))$. Compute $\boldsymbol{d} = \boldsymbol{c} - \boldsymbol{E}_{\boldsymbol{id}}^T\boldsymbol{p} \in \mathbb{Z}_q^l$. Output the plaintext $\boldsymbol{b} \in \mathbb{Z}_p^l$ by computing decode$(\boldsymbol{d})$.

They define admissible hash functions as a variant of the definition from [BB04]. For the details, see the original papers [BB04, CHK09]. Cash et al. showed the following security results.

Theorem 14.4.2 ([CHK09])**.** *Let $q \geq 5 \cdot s_d \cdot (m + l)$, $\chi = \bar{\Psi}_\alpha$, $1/\alpha \geq s_d \cdot \sqrt{(\lambda d + 1)m + l} \cdot \omega(\sqrt{\log n})$. If $\mathcal{H}_n$ is collision resistant, then the above HIBE is* ind-sID-cpa *secure. If $\mathcal{H}_n$ is a family of admissible hash functions, then the above HIBE is* ind-fID-cpa *secure.*

14.4.2 Another Scheme

Cash et al. [CHK09] also proposed the HIBE secure in the random oracle model. This scheme can be considered as the direct generalization of GPV-HIBE.

Scheme 14.4.3 (CHK-HIBEinROM [CHK09])**.** We model $G : \{0,1\}^* \to \mathbb{Z}_q^n$ and $H : \{0,1\}^* \to \mathbb{Z}_q^{n\times m}$ as the random oracles.

Setup(n)**:** On input the security parameter n, invoke the trapdoor algorithm LPSF.TrapGen(1^n) in Chapter 10 and obtain $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times m}$ and its basis $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ such that $\|\tilde{\boldsymbol{T}}\| \le \tilde{L}$. Output $param = \boldsymbol{A}_0$ and $msk = \boldsymbol{T}$.

Ext($param = \boldsymbol{A}_0, msk = \boldsymbol{T}, \boldsymbol{id}$)**:** Define $\boldsymbol{A}_{\boldsymbol{id}} = [\boldsymbol{A}_0|\boldsymbol{A}_1|\dots|\boldsymbol{A}_k] \in \mathbb{Z}_q^{n\times(k+1)m}$, where $\boldsymbol{A}_i \leftarrow H(\boldsymbol{id}, i) \in \mathbb{Z}_q^{n\times m}$, and $\boldsymbol{u}_{\boldsymbol{id}} \leftarrow G(\boldsymbol{id}) \in \mathbb{Z}_q^n$. Using a short basis $\boldsymbol{T}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_0)$, it samples a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}})$ and $\boldsymbol{e}_{\boldsymbol{id}} \leftarrow$ LPSF.SamplePre($\boldsymbol{A}_{\boldsymbol{id}|k-1}, \boldsymbol{T}_{\boldsymbol{id}|k-1}, s(k), \boldsymbol{u}_{\boldsymbol{id}}$). Output $dk_{\boldsymbol{id}} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{e}_{\boldsymbol{id}})$.

Delg($param = \boldsymbol{A}_0, usk_{\boldsymbol{id}|k-1} = (\boldsymbol{T}_{\boldsymbol{id}|k-1}, \boldsymbol{e}_{\boldsymbol{id}|k-1}), \boldsymbol{id}$)**:** $usk_{\boldsymbol{id}} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{e}_{\boldsymbol{id}})$. Define $\boldsymbol{A}_{\boldsymbol{id}}$ and $\boldsymbol{u}_{\boldsymbol{id}}$ as in the above. Using a short basis $\boldsymbol{T}_{\boldsymbol{id}|k-1}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}|k-1})$, it samples a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}})$ and $\boldsymbol{e}_{\boldsymbol{id}} \leftarrow$ LPSF.SamplePre($\boldsymbol{A}_{\boldsymbol{id}|k-1}, \boldsymbol{T}_{\boldsymbol{id}|k-1}, s(k), \boldsymbol{u}_{\boldsymbol{id}}$). Output $dk_{\boldsymbol{id}} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{e}_{\boldsymbol{id}})$.

Enc($param, id, msg = w \in \mathbb{Z}_p$)**:** First generate $\boldsymbol{A}_{\boldsymbol{id}|k-1}$ as in the above. Then, generate $\boldsymbol{s} \leftarrow \mathbb{Z}_q^n$ and $\boldsymbol{x} \leftarrow \chi^{km}$. Compute $\boldsymbol{p} = \boldsymbol{A}_{\boldsymbol{id}|k-1}^T \boldsymbol{s} + \boldsymbol{x} \in \mathbb{Z}_q^{km}$. Compute $c = \boldsymbol{u}_{\boldsymbol{id}}^T \boldsymbol{s} + x' + \mathsf{encode}(w)$, where $\boldsymbol{x} \leftarrow \chi$. Then, the ciphertext is $(\boldsymbol{p}, c)$.

Dec($usk_{id} = (\boldsymbol{T}_{\boldsymbol{id}}, \boldsymbol{e}_{\boldsymbol{id}}), ct = (\boldsymbol{p}, c)$) . Compute $d = c - \boldsymbol{e}_{\boldsymbol{id}}^T \boldsymbol{p} \in \mathbb{Z}_q$. Output the plaintext $w \in \mathbb{Z}_p$ by computing decode(d).

14.5 Peikert's "Bonsai" Key-Encapsulation Mechanism

Peikert also proposed hierarchical identity-based encryption scheme. This scheme can be considered as optimized variant of the CHK-HIBE.

First, if each component id_i of $\boldsymbol{id}$ is restricted to λ-bit, we have no need to introduce the hash function H_i (because the identity map is collision resistant). Second, using LWE-KEM = (K.Gen, K.Encaps, K.Decaps) (Section 13.4), $\boldsymbol{e}_{\boldsymbol{id}}$ is eliminated since the basis $\boldsymbol{T}_{\boldsymbol{id}}$ suffices to decrypt. Third, he and Alwen improved the trapdoor generation (the third construction in Section 10.3.4).

14.5.1 Descriptions

Scheme 14.5.1 (Bonsai-HIBKEM [Pei09b])**.** Suppose that the maximal depth is d. Let $m = m_1 + m_2$.

Setup(n)**:** On input the security parameter n, invoke the trapdoor algorithm LPSF.TrapGen(1^n) in Chapter 10 and obtain $\boldsymbol{A}_0 \in \mathbb{Z}_q^{n\times m}$ and its basis $\boldsymbol{T} \in \mathbb{Z}^{m\times m}$ such that $\|\tilde{\boldsymbol{T}}\| \le L$. Next, generate random matrices $\boldsymbol{C}_{i,j}^{(b)} \leftarrow \mathbb{Z}_q^{n\times m_2}$ for

$i \in [d]$, $j \in [\lambda]$ and $b \in \{0,1\}$. Additionally, choose $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_l] \leftarrow \mathbb{Z}_q^{n \times l}$. Output $param = (\boldsymbol{A}_0, \boldsymbol{U}, \{\boldsymbol{C}_{i,j}^{(b)}\}, \{H_i\})$ and $msk = \boldsymbol{T}$.

Ext($param = \boldsymbol{A}_0, msk = \boldsymbol{T}, \boldsymbol{id}$)**:** For an identity $\boldsymbol{id} = (id_1, \ldots, id_k)$, define $\boldsymbol{A}_{\boldsymbol{id}} = [\boldsymbol{A}_0 | \boldsymbol{A}_{1,id_1} | \ldots | \boldsymbol{A}_{k,id_k}]$ where $\boldsymbol{A}_{i,id_i} \leftarrow [\boldsymbol{C}_{i,1}^{(t_1)} | \ldots | \boldsymbol{C}_{i,\lambda}^{(t_\lambda)}] \in \mathbb{Z}_q^{n \times \lambda m_2}$ for $t = id_i$. Using a short basis $\boldsymbol{T}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_0)$, it samples a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}})$. Output $dk_{\boldsymbol{id}} = \boldsymbol{T}_{\boldsymbol{id}}$.

Delg($param = \boldsymbol{A}_0, usk_{\boldsymbol{id}|k-1} = \boldsymbol{T}_{\boldsymbol{id}|k-1}, \boldsymbol{id}$)**:** Define $\boldsymbol{A}_{\boldsymbol{id}}$ as in the above. Using a short basis $\boldsymbol{T}_{\boldsymbol{id}|k-1}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}|k-1})$, it construct a short basis $\boldsymbol{T}'$ of the lattice $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}})$. Then, it samples a basis $\boldsymbol{T}_{\boldsymbol{id}}$ of $\Lambda_q^{\perp}(\boldsymbol{A}_{\boldsymbol{id}})$. Output $dk_{\boldsymbol{id}} = \boldsymbol{T}_{\boldsymbol{id}}$.

Enc($param, \boldsymbol{id}$)**:** It outputs $(\boldsymbol{k}, \sigma) \leftarrow$ K.Encaps($\boldsymbol{A}_{\boldsymbol{id}}$).

Dec($\boldsymbol{T}_{\boldsymbol{id}}, \sigma$)**:** It outputs $\boldsymbol{k} \leftarrow$ K.Decaps($\boldsymbol{T}_{\boldsymbol{id}}, \sigma$).

Remark 14.5.2. Using the miniature "Bonsai" techniques, we can obtain the ideal-lattice-based IBE and HIBE as in [SSTX09].

15 Proxy Re-Encryption

Proxy re-encryption enables a proxy to convert a ciphertext for some user to a ciphertext for another user, but a proxy cannot learn information of messages. All of the proxy re-encryption and identity-based proxy re-encryption schemes are based on the number-theoretic assumptions. This paper proposed proxy re-encryption schemes based on the learning with errors problem. They are first schemes based on combinatorial problems.

Organization: Section 15.1 gives the brief introduction of proxy re-encryption, gives the idea from the ElGamal-based proxy re-encryption scheme. Section 15.2 defines model and the security notions on proxy re-encryption. Section 15.3 studies the Xagawa–Tanaka proxy re-encryption scheme, which adds feature to Regev's encryption scheme. Section 15.4 also studies the variant of the above scheme.

15.1 Introduction

Suppose that Alice wants to forward a received encrypted e-mail to Bob in the public channel. She decrypts it by her secret key, encrypts the message with Bob's public key, and sends it to him. However, decryption and encryption are costly for her mobile phone in general. Therefore, she wants a mail server to forward her mail to Bob automatically. In this case, she does not trust the server, hence, she does not want to give her secret key to the server. The one of solutions is proxy re-encryption [BBS98].

In a proxy re-encryption (PRE) scheme, the server is given a re-encryption key $rk_{A\leftrightarrow B}$ between Alice and Bob. The server, given a ciphertext ct_A for Alice, can convert it to a ciphertext ct_B for Bob by using the re-encryption key $rk_{A\leftrightarrow B}$ and without decrypting ct_A. In addition, proxy re-encryption ensures that even if the server knows $rk_{A\leftrightarrow B}$, it cannot learn the message of ct_A.

The study of proxy re-encryption is initiated by Blaze, Bleumer, and Strauss [BBS98]. They formalize a proxy re-encryption and gave an example based on the ElGamal encryption scheme. There are several proxy re-encryption schemes [BBS98, AFGH06, CH07, LV08, DWLC08, ABH09, MNT10] and identity-based proxy re-encryption schemes [Mat07, GA07, CT07] in the literature. However, their underlying problems are the decisional Diffie-Hellman problem or its variants.

In this paper, we propose proxy re-encryption schemes based on other problems, the learning with errors and lattice problems. Our constructions are obtained by extending Regev's encryption scheme [Reg09].

Ideas from the ElGamal-based PRE: We note that some lattice-based cryptosystems have similar structure on the DDH-based cryptosystems while inherent noises of lattice-based cryptosystems disturb the structure.

Consider the ElGamal encryption scheme over $\mathbb{G} = \langle g \rangle$ with order a large prime q. The key pair is $(x, y = g^x)$ for randomly chosen x. The ciphertext of $w \in \mathbb{G}$ under the encryption key y is $(g^k, w \cdot y^k)$ for randomly chosen k. Let $(x_A, y_A = g^{x_A})$ and $(x_B, y_B = g^{x_B})$ denote Alice's and Bob's key pair, respectively. Assume that the proxy has the re-encryption key $r_{A\leftrightarrow B} = x_A - x_B$ and has the ciphertext (c_1, c_2) to be converted. Then, the conversion is done by

$$\begin{aligned}(c_1', c_2') &= (c_1, c_2 \cdot c_1^{-r_{A\leftrightarrow B}}) \\ &= (g^k, w \cdot g^{kx_A} \cdot g^{k(x_B - x_A)}) = (g^k, w \cdot y_B^k).\end{aligned}$$

It can be shown that this proxy re-encryption scheme is based on the hardness of the DDH problem.[1]

We here recall Regev's encryption scheme. The key pair is computed by $(\boldsymbol{s}, (\boldsymbol{A}, \boldsymbol{p} = \boldsymbol{s}^T\boldsymbol{A} + \boldsymbol{x}))$, where $\boldsymbol{s} \in \mathbb{Z}_q^n$, $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$, $\boldsymbol{x} \in \mathbb{Z}_q^{1\times m}$ and the magnitudes of the elements of $\boldsymbol{x}$ are relatively smaller than $q/4m$, say the l_1-norm of $\boldsymbol{x}$ is at most $q/4$. The encryption of the message $w \in \{0,1\}$ under the encryption key $(\boldsymbol{A}, \boldsymbol{p})$ is $(\boldsymbol{u}, v) = (\boldsymbol{A}\boldsymbol{e}, \boldsymbol{p}\boldsymbol{e} + w\lfloor q/2 \rfloor)$, where $\boldsymbol{e} \leftarrow \{0,1\}^m$.

The decryption procedure is as follows: (1) compute $d = v - \boldsymbol{s}^T\boldsymbol{u}$ and (2) output 0 if the absolute value of d is at most $q/4$ and output 1 otherwise.

Let $(\boldsymbol{s}_A, (\boldsymbol{A}_A, \boldsymbol{p}_A = \boldsymbol{s}_A^T\boldsymbol{A}_A + \boldsymbol{x}_A))$, and $(\boldsymbol{s}_B, (\boldsymbol{A}_B, \boldsymbol{p}_B = \boldsymbol{s}_B^T\boldsymbol{A}_B + \boldsymbol{x}_B))$ denote Alice's and Bob's key pair, respectively. Let $\boldsymbol{r}_{A\leftrightarrow B} = \boldsymbol{s}_A - \boldsymbol{s}_B$. Then, the conversion from $(\boldsymbol{u}, v_A)$ to $(\boldsymbol{u}, v_B)$ is done by $(\boldsymbol{u}, v_B) = (\boldsymbol{u}, v_A - \boldsymbol{r}_{A\leftrightarrow B}^T\boldsymbol{u})$, which is similar to that of the ElGamal-based proxy re-encryption scheme. The decryption by Bob works correctly since

$$d_B = v_B - \boldsymbol{s}_B^T\boldsymbol{u} = v_A - (\boldsymbol{s}_A - \boldsymbol{s}_B)^T\boldsymbol{u} - \boldsymbol{s}_B^T\boldsymbol{u} = v_A - \boldsymbol{s}_A^T\boldsymbol{u} = d_A.$$

[1] In the BBS scheme [BBS98], the re-encryption key is $r_{A\leftrightarrow B} = x_A/x_B$. The ciphertext of w is $(c_1, c_2) = (w \cdot g^k, y_A^k)$. The conversion is done by $(c_1', c_2') = (c_1, c_2^{1/r_{A\leftrightarrow B}}) = (w \cdot g^k, y_B^k)$.

The proof strategy for security is also similar to that of the ElGamal-based proxy re-encryption scheme.

The leftover hash lemma often appears in the context of lattice-based cryptography. We summarize the arguments which appeared in many papers on lattice-based cryptography. See [Reg09] for the proof.

Lemma 15.1.1 (The uniformity lemma for lattice-based hash functions)**.** *Consider* $\mathcal{H} = \{h_{\boldsymbol{A}} : \{0,1\}^m \to \mathbb{Z}_q^{n+l} \mid \boldsymbol{A} \in \mathbb{Z}_q^{(n+l)\times m}\}$*, where* $h_{\boldsymbol{A}}(\boldsymbol{e}) = \boldsymbol{A}\boldsymbol{e}$*. Let* H *be the uniform distribution over* $\mathcal{H}$*, and* X *and* U *random variables distributed uniformly over* $\{0,1\}^m$ *and* $\mathbb{Z}_q^{n+l}$*, respectively. Applying the variant of the leftover hash lemma, we have*

$$\Pr_H[\Delta(H(X), U) \geq 2^{-\frac{1}{4}(m-(n+l)\log q)}] \leq 2^{-\frac{1}{4}(m-(n+l)\log q)}.$$

In particular, if $m = ((1+\delta)n + l)\log q$*, then we have that*

$$\Pr_H[\Delta(H(X), U) \geq q^{-\delta n/4}] \leq q^{-\delta n/4}.$$

15.2 Definitions

In this paper, we consider *bidirectional* and *multi-hop* proxy re-encryption. A PRE scheme is called bidirectional, if a proxy has a re-encryption key $rk_{i\leftrightarrow j}$, it can convert a ciphertext for the user i to a ciphertext for the user j, vice versa. A PRE scheme is said to be multi-hop, a proxy can re-encrypt a ciphertext for the user i into a ciphertext for the user j and it can re-encrypt that into one for the user k and so on.

15.2.1 Model of Proxy Re-Encryption Schemes

A PRE scheme PRE is a sextuplet of algorithms:

$\mathsf{Setup}(1^n)$**:** The setup algorithm, given the security parameter n, outputs parameters $param$.

$\mathsf{Reg}(param, i)$**:** The registration algorithm, given the parameters $param$ and a user identity i, outputs the pair of an encryption key and a decryption key (ek_i, dk_i).

$\mathsf{ReKeyGen}(dk_i, dk_j)$**:** The re-encryption key generation algorithm, given two decryption keys dk_i and dk_j, outputs a re-encryption key $rk_{i,j}$.

$\mathsf{Enc}(param, ek_i, msg)$**:** The encryption algorithm, given the parameters $param$, the encryption key ek_i of the user i, and a message msg, outputs a ciphertext ct_i.

$\mathsf{ReEnc}(rk_{i,j}, ct_i)$**:** The re-encryption algorithm, given the re-encryption key $rk_{i,j}$ between the users i and j, and a ciphertext ct_i for the user i, it outputs a ciphertext ct_j for the user j.

$\mathsf{Dec}(dk, ct)$**:** The decryption algorithm, given the decryption key dk and the ciphertext ct, outputs a plaintext msg.

Our definition of correctness is slightly weaker than the standard one [CH07]. We say a PRE scheme PRE is correct if an underlying public-key encryption scheme $\mathsf{PKE} = (\mathsf{Setup}, \mathsf{Reg}, \mathsf{Enc}, \mathsf{Dec})$ is correct. Formally, it holds that if for any valid msg, there exists some negligible function $\mathsf{negl}(n)$ such that for any i

$$\Pr\left[msg \neq \widetilde{msg} : \begin{array}{l} param \leftarrow \mathsf{Setup}(1^n); \\ (ek_i, dk_i) \leftarrow \mathsf{Reg}(param, i); \\ ct \leftarrow \mathsf{Enc}(param, ek_i, msg); \\ \widetilde{msg} \leftarrow \mathsf{Dec}(dk_i, ct); \end{array} \right] \leq \mathsf{negl}(n).$$

Additionally, we say a PRE scheme PRE is multi-hop correct if for any valid msg and for any integer $k > 1$, one can correctly decrypt the ciphertext of msg converted k times into msg, that is,

$$\Pr\left[msg \neq \widetilde{msg} : \begin{array}{l} param \leftarrow \mathsf{Setup}(1^n); \\ (ek_i, dk_i) \leftarrow \mathsf{Reg}(param, i); \\ rk_{i \leftrightarrow i+1} \leftarrow \mathsf{ReKeyGen}(dk_i, dk_{i+1}); \\ ct_1 \leftarrow \mathsf{Enc}(param, ek_1, msg); \\ ct_{i+1} \leftarrow \mathsf{ReEnc}(rk_{i \leftrightarrow i+1}, ct_i); \\ \widetilde{msg} \leftarrow \mathsf{Dec}(dk_k, ct_k); \end{array} \right] \leq n^{-\omega(1)},$$

where i runs from 1 to k.

15.2.2 Security Notions

We describe the formal definition of CPA security of proxy re-encryption, denoted by IND-PRE-CPA. Consider the following experiment $\mathbf{Exp}^{\text{ind-pre-cpa}}_{\mathsf{PRE},\mathcal{A}}(n)$ between the challenger C and the adversary $\mathcal{A}$.

Setup Phase: The challenger takes a security parameter n. It sets $HU, CU \leftarrow \emptyset$, runs the algorithm Setup with 1^n, and obtains parameters $param$, where HU and CU denote the sets of honest users and corrupted users, respectively. It gives $\mathcal{A}$ the parameters $param$.

Challenge Phase: In this phase, the adversary issues queries to the following oracles in any order and many times except to the constraint in the oracle Challenge.

- The oracle Init receives an index i. If $i \in HU \cup CU$ then it returns $\perp$. Otherwise, it obtains $(ek_i, dk_i) \leftarrow \mathsf{Reg}(param, i)$, adds i to HU, and provides $\mathcal{A}$ with ek_i.
- The oracle Corr receives an index i. If $i \in HU \cup CU$ then it returns $\perp$. Otherwise, it generates $(ek_i, dk_k) \leftarrow \mathsf{Reg}(param; r_i)$, adds i to CU, and provides $\mathcal{A}$ with (ek_i, dk_i) and r_i.

- The oracle ReKey receives two indices $i, j \in HU \cup CU$. If $i, j \in HU$ or $i, j \in CU$ returns $rk_{i \leftrightarrow j} \leftarrow \mathsf{ReKeyGen}(dk_i, dk_j)$. Otherwise, the oracle returns $\perp$.
- The oracle ReEnc receives two indices $i, j \in HU \cup CU$ and a ciphertext ct. If $i, j \in HU$ or $i, j \in CU$, then it obtains $rk_{i \leftrightarrow j} \leftarrow \textsc{ReKey}(dk_i, dk_j)$, obtains $\widetilde{ct} \leftarrow \mathsf{ReEnc}(param, rk_{i \leftrightarrow j}, ct)$, and provides $\mathcal{A}$ with the new ciphertext $\widetilde{ct}$. Otherwise, the oracle returns $\perp$.
- The oracle Challenge can be queried only once. This oracle receives two plaintexts msg_0, msg_1 and a target user i^*. If i^* is not in HU then it provides $\perp$ with the challenger and C outputs 0 and halts. Otherwise, the oracle flips a coin $b \in \{0, 1\}$, sets the target ciphertext to be $ct^* \leftarrow \mathsf{Enc}(ek_{i^*}, msg_b)$, and sends ct^* to the adversary and b to the challenger.

Guessing Phase: Finally, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$. If $b' = b$, the challenger outputs 1, otherwise 0.

Definition 15.2.1 (IND-PRE-CPA security)**.** Let PRE be a PRE scheme, $\mathcal{A}$ an adversary, and n a security parameter. We define the advantage of $\mathcal{A}$ as

$$\mathbf{Adv}^{\text{ind-pre-cpa}}_{\mathsf{PRE},\mathcal{A}}(n) = \left| 2 \Pr\left[\mathbf{Exp}^{\text{ind-pre-cpa}}_{\mathsf{PRE},\mathcal{A}}(n) = 1 \right] - 1 \right|.$$

We say that PRE is IND-PRE-CPA secure if $\mathbf{Adv}^{\text{ind-pre-cpa}}_{\mathsf{PRE},\mathcal{A}}(\cdot)$ is negligible for every polynomial-time adversary $\mathcal{A}$.

Since we only consider IND-PRE-CPA security, we prohibit the adversary to re-encrypt ciphertexts from an honest user to a corrupted user. This is because that this access can simulates a decryption oracle of the honest user.

15.3 The Xagawa–Tanaka Proxy Re-Encryption Scheme

We employ the variant by Peikert, Vaikuntanathan, and Waters [PVW08] of Regev's public-key encryption scheme [Reg09]. The main algorithms are the same as those in the PVW scheme. We add to it a re-encryption key generation algorithm and a re-encryption algorithm appeared in Section 1.

15.3.1 Description

Our PRE scheme LWEPRE is defined as follows:

$\mathsf{Setup}(1^n)$**:** Given a security parameter n, it outputs $\perp$ as $param$.

$\mathsf{Reg}(\perp, i)$**:** It generates $\boldsymbol{A}_i \leftarrow \mathbb{Z}_q^{n \times m}$, $\boldsymbol{S}_i \leftarrow \mathbb{Z}_q^{n \times l}$, and $\boldsymbol{X}_i \leftarrow \chi^{l \times m}$, and computes $\boldsymbol{P}_i = \boldsymbol{S}_i^T \boldsymbol{A}_i + \boldsymbol{X}_i \in \mathbb{Z}_q^{l \times m}$. It outputs $ek_i = (\boldsymbol{A}_i, \boldsymbol{P}_i)$ and $dk_i = \boldsymbol{S}_i$.

$\mathsf{ReKeyGen}(dk_i = \boldsymbol{S}_i, dk_j = \boldsymbol{S}_j)$**:** It outputs $\boldsymbol{R}_{i \leftrightarrow j} = \boldsymbol{S}_i - \boldsymbol{S}_j \in \mathbb{Z}_q^{n \times l}$.

$\mathsf{Enc}(ek = (\boldsymbol{A}, \boldsymbol{P}), \boldsymbol{w})$**:** The message space is $\mathbb{Z}_p^l$. It, given $\boldsymbol{w}$, computes $\boldsymbol{t} = t(\boldsymbol{w}) \in \mathbb{Z}_q^l$, where $t(w) = \lfloor wq/p \rceil \in \mathbb{Z}_q$ and chooses a vector $\boldsymbol{e} \leftarrow \{0,1\}^m \subset \mathbb{Z}_q^m$ uniformly at random. It outputs a pair $(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{Ae}, \boldsymbol{Pe} + \boldsymbol{t}) \in \mathbb{Z}_q^n \times \mathbb{Z}_q^l$ as a ciphertext.

$\mathsf{ReEnc}(rk_{i\leftrightarrow j} = \boldsymbol{R}_{i\leftrightarrow j}, (\boldsymbol{u}, \boldsymbol{v}_i))$**:** It computes $\boldsymbol{v}_j = \boldsymbol{v}_i - \boldsymbol{R}_{i\leftrightarrow j}^T \boldsymbol{u}$ and outputs $(\boldsymbol{u}, \boldsymbol{v}_j)$.

$\mathsf{Dec}(dk = \boldsymbol{S}, (\boldsymbol{u}, \boldsymbol{v}))$**:** It computes $\boldsymbol{d} = \boldsymbol{v} - \boldsymbol{S}^T \boldsymbol{u} \in \mathbb{Z}_q^l$ and outputs the plaintext $\boldsymbol{w} \in \mathbb{Z}_p^l$ such that $\boldsymbol{d} - t(\boldsymbol{v}) \in \mathbb{Z}_q^l$ is closest to $\boldsymbol{0}$.

We add ReKeyGen and ReEnc to the variant of Regev's encryption scheme by Peikert, Vaikuntanathan, and Waters [PVW08]. The parameters setting for correctness appeared in [PVW08].

Theorem 15.3.1 (Correctness [PVW08])**.** *Let* $\chi = \bar{\Psi}_\alpha$. *Let* $q \geq 4pm$, *let* $\alpha \leq 1/(p\sqrt{m} \cdot g(n))$ *for any* $g(n) = \omega(\sqrt{\log n})$. *Then, the above scheme is correct.*

The multi-hop correctness is easily derived by the correctness.

Theorem 15.3.2 (Multi-hop correctness)**.** *Let* q, α, *and* g *be as in the above. Then, the above scheme is multi-hop correct.*

Proof. Consider the users $1, \dots, k$. Suppose that $(\boldsymbol{u}, \boldsymbol{v}_1)$ is the valid ciphertext under the encryption key $(\boldsymbol{A}_1, \boldsymbol{P}_1)$ of the user 1 and the re-encryption procedure is performed from 1 to k through $2, \dots, k-1$. By the re-encryption procedures, we have that

$$\boldsymbol{v}_k = \boldsymbol{v}_1 - \sum_{i=1}^{k-1} \boldsymbol{R}_{i\leftrightarrow i+1}^T \boldsymbol{u} = \boldsymbol{v}_1 - \sum_{i=1}^{k-1} (\boldsymbol{S}_i - \boldsymbol{S}_{i+1})^T \boldsymbol{u} = \boldsymbol{v}_1 - (\boldsymbol{S}_1 - \boldsymbol{S}_k)^T \boldsymbol{u},$$

where $\boldsymbol{S}_i$ denotes the decryption key of the user i. In the decryption procedure by the user k, $\boldsymbol{d}_k$ is computed as follows:

$$\boldsymbol{d}_k = \boldsymbol{v}_k - \boldsymbol{S}_k^T \boldsymbol{u} = \boldsymbol{v}_1 - (\boldsymbol{S}_1 - \boldsymbol{S}_k)^T \boldsymbol{u} - \boldsymbol{S}_k^T \boldsymbol{u} = \boldsymbol{v}_1 - \boldsymbol{S}_1^T \boldsymbol{u}.$$

So, we have that $\boldsymbol{d}_k = \boldsymbol{d}_1$. Therefore, the multi-hop correctness follows from Theorem 15.3.1 straightforwardly. □

15.3.2 Security Proofs

The security of the scheme is based on the dLWE assumption.

Theorem 15.3.3 (Security)**.** *Let* $m \geq ((1+\delta)n + l)\log q$ *for* $\delta > 0$. *The above scheme is IND-PRE-CPA secure if* dLWE(q, χ) *is hard on average.*

Proof. It follows by combining the claims below. □

Sequence of games: We define the sequence of the games and bound the distance between the games.

Game_0**:** The original IND-PRE-CPA game. First, the challenger feeds $\perp$ to the adversary. The challenger simulates the oracles in the challenge phase. If the oracle CHALLENGE receives $(i^*, \boldsymbol{w}_0, \boldsymbol{w}_1)$, it flips a coin $b \in \{0,1\}$ and returns the target ciphertext $(\boldsymbol{u}^*, \boldsymbol{v}^*) = (\boldsymbol{A}_{i^*}\boldsymbol{e}^*, \boldsymbol{P}_{i^*}\boldsymbol{e}^* + t(\boldsymbol{w}_b))$, where $\boldsymbol{e}^* \leftarrow \{0,1\}^m$. Finally, the adversary outputs a guess b'. If $b = b'$, then the challenger outputs 1, otherwise 0.

Game_1**:** We modify the above game, by changing the generation methods of keys. At the beginning of the challenge phase, the challenger first generates re-encryption keys $\boldsymbol{R}_{1\leftrightarrow j} \leftarrow \mathbb{Z}_q^{n\times l}$ for $j = 2,\ldots,Q$. The other re-encryption key $\boldsymbol{R}_{i\leftrightarrow j}$ is computed by $\boldsymbol{R}_{i\leftrightarrow j} = \boldsymbol{R}_{1\leftrightarrow j} - \boldsymbol{R}_{1\leftrightarrow i}$. Next it chooses $\boldsymbol{S}_1 \leftarrow \mathbb{Z}_q^{n\times l}$, $\boldsymbol{A}_1 \leftarrow \mathbb{Z}_q^{n\times m}$, and $\boldsymbol{X}_1 \leftarrow \chi^{l\times m}$, and computes $\boldsymbol{P}_1 = \boldsymbol{S}_1^T\boldsymbol{A}_1 + \boldsymbol{X}_1$. If INIT is called with an input i, the challenger chooses $\boldsymbol{A}_i \leftarrow \mathbb{Z}_q^{n\times m}$, and $\boldsymbol{X}_i \leftarrow \chi^{l\times m}$, and computes $\boldsymbol{P}_i = \boldsymbol{S}_1^T\boldsymbol{A}_i - \boldsymbol{R}_{1\leftrightarrow i}^T\boldsymbol{A}_i + \boldsymbol{X}_i$. If REKEY is called with $i, j \in HU$, then it returns $\boldsymbol{R}_{i\leftrightarrow j}$. If REENC is called with $i, j, (\boldsymbol{u}, c)$, then it uses the re-encryption key $\boldsymbol{R}_{i\leftrightarrow j}$ to re-encrypt the ciphertext. The other conditions are the same as in the original game, Game_0.

Game_2**:** We replace the generation method of keys. The challenger queries to the oracle $A_{\boldsymbol{S},\chi}$ and obtains Qm samples $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{P}}) \in \mathbb{Z}_q^{n\times Qm} \times \mathbb{Z}_q^{l\times Qm}$. Then, it chops into $(\bar{\boldsymbol{A}}_i, \bar{\boldsymbol{P}}_i) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{l\times m}$ for $i = 1,\ldots,Q$. It sets $(\boldsymbol{A}_1, \boldsymbol{P}_1) = (\bar{\boldsymbol{A}}_1, \bar{\boldsymbol{P}}_1)$ and $(\boldsymbol{A}_i, \boldsymbol{P}_i) = (\bar{\boldsymbol{A}}_i, \bar{\boldsymbol{P}}_i - \boldsymbol{R}_{1\leftrightarrow i}^T\bar{\boldsymbol{A}}_i)$. The other conditions are the same as in the previous game, Game_1.

Game_3**:** We replace the oracle $A_{\boldsymbol{S},\chi}$ with $U(\mathbb{Z}_q^n \times \mathbb{Z}_q^l)$. Hence, the challenger obtains Qm samples $(\boldsymbol{A}, \boldsymbol{P})$ from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q^l)$ at first. Now, $\boldsymbol{P}$ is chosen uniformly at random.

Let S_i denote the event that the adversary wins, i.e., $b' = b$ in the game Game_i. We denote by $\mathbf{Adv}^{\mathrm{ind-pre-cpa}}_{\mathsf{LWEPRE},\mathcal{A}}(n)$ the advantage of the adversary $\mathcal{A}$ in the IND-PRE-CPA game with the security parameter n. By definition, we have that $\mathbf{Adv}^{\mathrm{ind-pre-cpa}}_{\mathsf{LWEPRE},\mathcal{A}}(n) = |2\Pr[S_0] - 1| = |\Pr[S_0] - \Pr[S_1]|$.

Claim 15.3.4. Game_0 *and* Game_1 *are identical.*

Proof. Recall that $\boldsymbol{R}_{i\leftrightarrow j} = \boldsymbol{S}_i - \boldsymbol{S}_j$ by the definition. Hence, we have that $\boldsymbol{R}_{i\leftrightarrow j} = \boldsymbol{R}_{1\leftrightarrow j} - \boldsymbol{R}_{1\leftrightarrow i}$ in Game_0. This calculation corresponds to the computation of $\boldsymbol{R}_{i\leftrightarrow j}$ in Game_1.

Additionally, in Game_1 we have $\boldsymbol{S}_i = \boldsymbol{S}_1 - \boldsymbol{R}_{1\leftrightarrow i}$ imaginary, since $\boldsymbol{P}_i = (\boldsymbol{S}_1 - \boldsymbol{R}_{1\leftrightarrow i})^T\boldsymbol{A}_i + \boldsymbol{X}_i$. Therefore, two games are identical. □

Claim 15.3.5. Game_1 *and* Game_2 *are identical.*

Proof. In Game_1, we have that $\boldsymbol{P}_i = \boldsymbol{S}_i^T\boldsymbol{A}_i + \boldsymbol{X}_i - \boldsymbol{R}_{1\leftrightarrow i}^T\boldsymbol{A}_i$.

In Game_2, we have that $\boldsymbol{P}_i = \bar{\boldsymbol{P}}_i - \boldsymbol{R}_{1\leftrightarrow i}^T\boldsymbol{A}_i$. Since the samples from $A_{\boldsymbol{S},\bar{\Psi}_\alpha}$ is $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{P}} = \boldsymbol{S}^T\bar{\boldsymbol{A}} + \boldsymbol{X})$, we conclude that two games are identical. □

Claim 15.3.6. *Game_2 and Game_3 are computationally indistinguishable if dLWE(q,χ) is hard on average.*

Proof. Notice that in both games, the challenger does not know the secret keys of the honest users. Hence, if the adversary $\mathcal{A}$ acts differently in Game_2 and Game_3, one can distinguish $A_{S,\chi}$ from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q^l)$ with Qm samples. This concludes the proof. □

Claim 15.3.7. *In Game_3, no adversary can obtain the information b if $m \geq ((1+\delta)n + l)\log q$. Formally, we have that*

$$\left|\Pr[S_3] - \frac{1}{2}\right| \leq 2q^{-\delta n/4} = \mathsf{negl}(n).$$

Proof. By the parameter setting, we can apply the leftover hash lemma to the target ciphertext and this concludes the proof. □

15.4 Extension

We next consider a variant of LWEPRE, denoted by $\mathsf{LWEPRE2}$. In this variant, users share $\boldsymbol{A}$ as the public parameter as users share the group $(\mathbb{G}, q, g)$ in the ElGamal encryption scheme.

$\mathsf{Setup}(n)$**:** Given input the security parameter n, it outputs a random matrix $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ as *param*.

$\mathsf{Reg}(\boldsymbol{A}, i)$**:** It generates $\boldsymbol{S}_i \leftarrow \mathbb{Z}_q^{n\times l}$, and $\boldsymbol{X}_i \leftarrow \bar{\Psi}_\alpha^{l\times m}$, and computes $\boldsymbol{P}_i = \boldsymbol{S}_i^T \boldsymbol{A} + \boldsymbol{X}_i \in \mathbb{Z}_q^{l\times m}$. It outputs $ek_i = \boldsymbol{P}_i$ and $dk_i = \boldsymbol{S}_i$.

$\mathsf{ReKeyGen}$**,** Enc**,** ReEnc**,** Dec**:** They are the same as in LWEPRE.

The correctness and the multi-hop correctness of $\mathsf{LWEPRE2}$ follow from these of LWEPRE. In order to show the security, we need a lemma on the Gaussian below.

Key Lemma: The following lemma states that the discretized folded Gaussian with variance $\alpha^2/2\pi$ statistically hides the discretized folded Gaussian with variance $\delta^2\alpha^2/2\pi$, when δ is negligible. The similar lemma appears in [Reg09, GKPV10]. Additionally, the lemmas are used to construct a key-leakage resilient secret-key encryption scheme [GKPV10] and a key-dependent-message secure public-key encryption scheme [BGK09].

Binding two following claims, our lemma is obtained.

Lemma 15.4.1. *Let $q = q(n)$ be super-polynomial integer function of n and $\alpha = \alpha(n) > 0$ and $\delta \in (0,1)$ reals. If δ is $n^{-\omega(1)}$, then the statistical distance between $\bar{\Psi}_\alpha$ and $\bar{\Psi}_\alpha + \bar{\Psi}_{\delta\alpha}$ is at most $n^{-\omega(1)}$.*

A similar claim already appeared in [Reg09, Claim 2.2], the statistical distance between Ψ_α and $\Psi_{(1+\delta)\alpha} = \Psi_\alpha + \Psi_{\delta\alpha}$ is at most 9δ for any $\delta \in [0,1)$, whose distributions are not discretized.

Proof. Let $\mu = \delta q \alpha t$ be a natural number. Then, from Claim 15.4.2, we have that $\Pr[|\bar{X}| \geq \mu]$ is at most $\frac{1}{\sqrt{2\pi}t}\exp(-\pi t^2)$. For $\mu \leq \mu'$, we have that the statistical distance between $\bar{\Psi}_\alpha$ and $\bar{\Psi}_\alpha + \mu'$ is at most $(\mu+2)/q\alpha$. Hence, the statistical distance between $\bar{\Psi}_\alpha$ and $\bar{\Psi}_\alpha + \bar{\Psi}_{\delta\alpha}$ is at most $\frac{1}{\sqrt{2\pi}t}\exp(-\pi t^2) + 2\delta t$. By setting $t = \omega(\sqrt{\log n}) \in \mathrm{poly}(n)$ and $\delta t = n^{-\omega(1)}$, we have that the upperbound is $n^{-\omega(1)}$. □

For example, we set $q(n) = n^{2\log n}$, $\alpha = 1/n^2$, $\delta = n^{-\log n}$, $t = \log n$. Then, $q \cdot \delta\alpha = n^{\Theta(\log n)}$ is super-polynomial in n and $\delta t = n^{-\Theta(\log n)}$ is negligible in n.

Claim 15.4.2. *Let $\bar{X}$ be a random variable according to $\bar{\Psi}_{\delta\alpha}$. Then,*

$$\Pr[|\bar{X}| \leq \mu] \geq 1 - B(q,\alpha,\delta,\mu),$$

where

$$B(q,\alpha,\delta,\mu) = \frac{\delta q \alpha}{(\mu+1/2)\sqrt{2\pi}} \cdot \exp\left(-\frac{\pi(\mu+1/2)^2}{\delta^2 q^2 \alpha^2}\right).$$

In particular, if $\mu = \delta q\alpha \cdot \omega(\sqrt{\log n})$, $\Pr[|\bar{X}| \geq \mu]$ is negligible in n.

Proof. Let $B_\delta = \frac{\delta q\alpha}{(\mu+1/2)\sqrt{2\pi}}\exp(-\pi(\mu+1/2)^2/\delta^2q^2\alpha^2)$. In order to prove the claim, it is sufficient to show that, for $X \sim \Psi_{\delta\alpha}$, $\Pr[|X| \geq (\mu+1/2)/q] \leq B(q,\alpha,\delta,\mu)$. Hence, we show that, for $X \sim N(0,(\delta\alpha)^2/2\pi)$, $\Pr[|X| \geq (\mu+1/2)/q] \leq B(q,\alpha,\delta,\mu)$.

Applying the tail bound for the Gaussian that $\Pr[|X| \geq t\sigma] \leq \frac{1}{t}\cdot\exp(-t^2/2)$ for $X \sim N(0,\sigma^2)$, we have that

$$\Pr[|X| \geq (\mu+1/2)/q] \leq \frac{\delta q\alpha}{(\mu+1/2)\sqrt{2\pi}} \cdot \exp\left(-\frac{\pi(\mu+1/2)^2}{\delta^2q^2\alpha^2}\right).$$

This completes the proof. □

Claim 15.4.3. *For any $\alpha > 0$, any $q \in \mathbb{N}$, and any $\mu \in \mathbb{N}$, the statistical distance between $\bar{\Psi}_\alpha$ and $\bar{\Psi}_\alpha + \mu$ is at most $(\mu+2)/q\alpha$.*

Proof. Let us consider a statistical distance Δ_μ between $dN_q(\alpha^2/2\pi)$ and $dN_q(\alpha^2/2\pi) + \mu$, where $dN_q(\sigma^2)$ is the following distribution; samples X from $N(0,\sigma^2)$ and returns $\lfloor qX \rceil$. Since $\Delta_\mu \geq \Delta(\bar{\Psi}_\alpha, \bar{\Psi}_\alpha + \mu)$, we bound this distance by $(\mu+2)/q\alpha$.

It is obvious that $\Delta_\mu \geq \Delta_{\mu'}$ if $\mu \geq \mu'$. Hence, we assume that μ is even and show that $\Delta_\mu \leq (\mu+1)/q\alpha$. Now, since μ is even, the probability that $\mu/2$ is the sample from $dN_q(\alpha^2/2\pi)$ equals to the probability that from $dN_q(\alpha^2/2\pi) + \mu$. Therefore, we have that

$$
\begin{aligned}
\Delta_\mu &\le 2 \sum_{k<\mu/2} \Pr_{X\sim dN_q(\alpha^2/2\pi)}[X=k] - \Pr_{X\sim dN_q(\alpha^2/2\pi)+\mu}[X=k] \\
&= 2 \sum_{k<\mu/2} \int_{k-1/2}^{k+1/2} \frac{1}{q\alpha}\rho_{q\alpha}(x)dx - \int_{k-1/2}^{k+1/2} \frac{1}{q\alpha}\rho_{q\alpha}(x-\mu)dx \\
&= 2\left(\int_{-\infty}^{\mu/2+1/2} \frac{1}{q\alpha}\rho_{q\alpha}(x)dx - \int_{-\infty}^{\mu/2+1/2} \frac{1}{q\alpha}\rho_{q\alpha}(x-\mu)dx\right) \\
&= \Pr_{X\sim N(0,q^2\alpha^2/2\pi)}[X \le \mu/2+1/2] \\
&\quad - \Pr_{X\sim N(0,q^2\alpha^2/2\pi)}[X \le -\mu/2+1/2] \\
&\le \Pr_{X\sim N(0,q^2\alpha^2/2\pi)}[|X| \le \mu/2+1/2] \\
&= \int_{-(\mu+1)/2}^{(\mu+1)/2} \frac{1}{q\alpha}\exp\left(-\pi\frac{x^2}{q^2\alpha^2}\right)dx \\
&\le \int_{-(\mu+1)/2}^{(\mu+1)/2} \frac{1}{q\alpha}dx = \frac{\mu+1}{q\alpha}.
\end{aligned}
$$

□

Proof of Security: We define the sequence of the games and bound the distance between the games.

Game$_0$: The original IND-PRE-CPA game. First, the challenger feeds $\boldsymbol{A} \leftarrow \mathbb{Z}_q^{n\times m}$ to the adversary $\mathcal{A}$. The challenger simulates the oracles in the challenge phase. If the oracle Challenge receives $(i^*, \boldsymbol{w}_0, \boldsymbol{w}_1)$, it flips a coin $b \in \{0,1\}$ and returns the target ciphertext $(\boldsymbol{u}^*, \boldsymbol{v}^*) = (\boldsymbol{A}\boldsymbol{e}^*, \boldsymbol{P}_{i^*}\boldsymbol{e}^* + t(\boldsymbol{w}_b))$, where $\boldsymbol{e}^* \leftarrow \{0,1\}^m$. Finally, the adversary outputs a guess b'. If $b = b'$, then the challenger outputs 1, otherwise 0.

Game$_1$: We modify the above game, by changing the generation methods of keys. At the beginning of the challenge phase, the challenger first generates re-encryption keys $\boldsymbol{R}_{1\leftrightarrow j} \leftarrow \mathbb{Z}_q^{n\times l}$ for $j = 2,\dots,Q$. The other re-encryption key $\boldsymbol{R}_{i\leftrightarrow j}$ is computed by $\boldsymbol{R}_{i\leftrightarrow j} = \boldsymbol{R}_{1\leftrightarrow j} - \boldsymbol{R}_{1\leftrightarrow i}$. Next it chooses $\boldsymbol{S}_1 \leftarrow \mathbb{Z}_q^{n\times l}$ and $\boldsymbol{X}_1 \leftarrow \chi^{l\times m}$, and computes $\boldsymbol{P}_1 = \boldsymbol{S}_1^T\boldsymbol{A} + \boldsymbol{X}_1$. If Init is called with an input i, the challenger chooses and $\boldsymbol{X}_i \leftarrow \chi^{l\times m}$, and computes $\boldsymbol{P}_i = \boldsymbol{S}_1^T\boldsymbol{A} - \boldsymbol{R}_{1\leftrightarrow i}^T\boldsymbol{A} + \boldsymbol{X}_i$. If ReKey is called with $i, j \in HU$, then it returns $\boldsymbol{R}_{i\leftrightarrow j}$. If ReEnc is called with $i, j, (\boldsymbol{u}, c)$, then it uses the re-encryption key $\boldsymbol{R}_{i\leftrightarrow j}$ to re-encrypt the ciphertext. The other conditions are the same as in the original game, Game$_0$.

Game$_{1.5}$: We change the generation method of the noises. We replace $\boldsymbol{X}_1,\dots,\boldsymbol{X}_Q \leftarrow \bar{\Psi}_\alpha^{l\times m}$ with $\boldsymbol{X}+\boldsymbol{X}_1,\dots,\boldsymbol{X}+\boldsymbol{X}_Q$, where $\boldsymbol{X} \leftarrow \bar{\Psi}_{\delta\alpha}^{l\times m}$. Hence, the key of the user i is $\boldsymbol{P}_i = \boldsymbol{S}_1^T\boldsymbol{A} - \boldsymbol{R}_{1\leftrightarrow i}^T\boldsymbol{A} + \boldsymbol{X} + \boldsymbol{X}_i$.

Game$_2$**:** We replace the key of the user 1. The challenger queries to the oracle $A_{\boldsymbol{S},\bar{\Psi}_{\delta\alpha}}$ and obtains m samples $(\bar{\boldsymbol{A}}, \bar{\boldsymbol{P}} = \boldsymbol{S}^T\bar{\boldsymbol{A}} + \bar{\boldsymbol{X}}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{l\times m}$. It computes $\boldsymbol{P}_i = \bar{\boldsymbol{P}} - \boldsymbol{R}_{1\leftrightarrow i}^T\bar{\boldsymbol{A}} + \boldsymbol{X}_i$, where $\boldsymbol{X}_i \leftarrow \bar{\Psi}_\alpha^{l\times m}$ for $i = 1, \ldots, k$. The other conditions are the same as in the previous game, Game$_{1.5}$.

Game$_3$**:** We replace the oracle $A_{\boldsymbol{S},\bar{\Psi}_{\delta\alpha}}$ with $U(\mathbb{Z}_q^n \times \mathbb{Z}_q^l)$. Then, the challenger obtains m samples $(\boldsymbol{A}, \boldsymbol{P})$ from $U(\mathbb{Z}_q^n \times \mathbb{Z}_q^l)$.

The main strategy of the security proof is similar to that in the previous one. We note that Game$_1$ and Game$_{1.5}$ is statistically identical if the parameter settings satisfy the conditions in Lemma 5.8. The other games are statistically or computationally identical as in the previous proofs. We omit the details since they are very similar to the previous proof.

15.5 Concluding Remarks

We remark that anyone can obtain the re-encryption key by using the proxy; Let us order the proxy to convert the ciphertext $(\boldsymbol{i}_k, \boldsymbol{0})$, where $k \in [n]$, for the user i to the user j. Then the proxy returns $(\boldsymbol{i}_k, -\boldsymbol{R}_{i\leftrightarrow j}^T\boldsymbol{i}_k)$. By repeating the conversion with $k = 1, \ldots, n$, we obtain $-\boldsymbol{R}_{i\leftrightarrow j}^T$, that is, the re-encryption key between i and j.

In the real world, this can be considered as an attack. However, the IND-PRE-CPA security does not capture this attacks. Hence, we should define the security on leaks of the re-encryption keys in the CPA settings. We finally note that the IND-PRE-CCA security captures this attacks, see [CH07, MNT10].

Acknowledgement

I would like to thank my supervisor Keisuke Tanaka for insightful suggestions and creative comments. Without his guidance and wide knowledge, this thesis could not be done. I would also like to thank Akinori Kawachi for thoughtful discussions and supports, and the past that he have dragged me into lattice-based cryptography. I want to thank you Kenji Yasunaga for several fruitful discussions.

I would like to thank all members of Tanaka Laboratory for their advice, encouragement, friendship, and enjoyable and delightful non-working hours. Thank you, Keiji Omura, Ryoutaro Hayashi, Toshiyuki Isshiki, Akihiro Mihara, Takao Onodera, Hiroki Hada, Manabu Suzuki, Naoyuki Yamashita, Shizu Kanauchi, Takato Hirano, Christopher Alfred Portman, Harugana Hiwatari, Jun Nakajima, Chihiro Ohyama, Masatoshi Yashiro, Ryo Nishimaki, Mario Larangeira Junior, Kouich Sakumoto, Tatsunori Seki, Hirotoshi Takebe, Daisuke Inoue, Akira Numayama, Koichiro Wada, Hideaki Suzuki, Toshihide Matsuda, Yuki Tan, Hinako Kamimura, Hitoshi Namiki, Akihiro Yamada, Manh Ha Nguyen, Chiaki Minato, and Hapuarachchillage D. P. S. S. Kumara.

I am thankful to Osamu Watanabe for giving me a chance to intern in NTT. I would like to express deep gratitude for the members of NTT Information Sharing Platform Laboratories. I especially thank to Eiichiro Fujisaki, the mentor in NTT, for his encouragements and fruitful discussions in the summer intern.

It gives me pleasure to thank you friends, Koji Takasu, Kazuhiro Shimmura, and You Koseki. Thank you Yasuhito Higa, Youichi Kawanishi (as known as Kirino), Yoshitsugu Kitagawa, and Keiichi Sakamoto to drag me into dangerous games and spend delightful days. I have spent many non-working hours with my room mates. I thank you, Yasuhito Higa, Alkin, Akira Tsurushima, Keiichi Sakamoto, Futoshi Shirose, Hideaki Takahasi, Nazuna Tsuchida, Kouichiro Hotta, Atsushi Takada, and Mayumi Kawamura. Finally, I would like to thank my family for their kindness and financial supports.

I note that some parts of this thesis are supported by KAKENHI No.19-55201.

References

[AABN02] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Minimizing assumptions for security and forward-security. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 418–433. Springer-Verlag, 2002.

[AABN08] Michel Abdalla, Jee Hea An, Mihir Bellare, and Chanathip Namprempre. From identification to signatures via the Fiat-Shamir transform: Necessary and sufficient conditions for security and forward-security. *IEEE Transactions on Information Theory*, 54(8):3631–3646, 2008.

[AC02] Mark Adcock and Richard Cleve. A quantum Goldreich-Levin theorem with cryptographic applications. In Helmut Alt and Afonso Ferreira, editors, *STACS 2002*, volume 2285 of *Lecture Notes in Computer Science*, pages 323–334. Springer-Verlag, 2002. [arXiv:quant-ph/0108095].

[AB09] Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. Manuscript, July 2009. Available at `http://www.cs.stanford.edu/~xb/ab09/`.

[AEVZ02] Erik Agrell, Thomas Eriksson, Alexander Vardy, and Kenneth Zeger. Closest point search in lattices. *IEEE Transactions on Information Theory*, 48(8):2201–2214, 2002.

[AMGH08] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additive homomorphic encryption with t-operand multiplications. Cryptology ePrint Archive, Report 2008/378, 2008.

[AR05] Dorit Aharonov and Oded Regev. Lattice problems in NP cap coNP. *Journal of the ACM*, 52(5):749–765, 2005.

[Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings on 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 99–108, Philadelphia, Pennsylvania, USA, May 1996. ACM. See also ECCC TR96-007.

[Ajt98] Miklós Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions (extended abstract). In *Proceedings on 30th Annual ACM Symposium on Theory of Computing (STOC '98)*, pages 10–19. ACM, 1998. See also ECCC TR97-047.

[Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In Jirí Wiedermann, Peter van Emde Boas, and Mogens Nielsen, editors, *ICALP '99*, volume 1644 of *Lecture Notes in Computer Science*, pages 1–9, 1999.

[AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *Proceedings on 29th Annual ACM Symposium on Theory of Computing (STOC '97)*, pages 284–293. ACM, 1997. See also ECCC TR96-065.

[AKS01] Miklós Ajtai, S.-Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing (STOC 2001)*, pages 601–610. ACM, 2001.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer-Verlag, 2009.

[AKKV05] Mikhail Alekhnovich, Subhash Khot, Guy Kindler, and Nisheeth K. Vishnoi. Hardness of approximating the closest vector problem with pre-processing. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2005)*, pages 216–225. IEEE Computer Society, 2005.

[AP09] Joël Alwen and Chris Peikert. Generating shorter bases for hard random lattices. In Susanne Albers and Jean-Yves Marion, editors, *STACS 2009*, volume 3 of *LIPIcs*, pages 75–86, Germany, 2009. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer-Verlag, 2009.

[ABSS97] Sanjeev Arora, László Babai, Jacques Stern, and Z. Sweedyk. The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences*, 54(2):317–331, 1997.

[ABH09] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. Key-private proxy re-encryption. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 279–294. Springer-Verlag, 2009.

[AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security (TISSEC)*, 9(1):1–30, February 2006.

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

[Ban95] Wojciech Banaszczyk. Inequalites for convex bodies and polar reciprocal lattices in $\mathbb{R}^n$. *Discrete & Computational Geometry*, 13:217–231, 1995.

[BHHI09] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. Cryptology ePrint Archive, Report 2009/511, 2009.

[BDJR97] Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 394–403. IEEE Computer Society, 1997.

[BHK09] Mihir Bellare, Dennis Hofheinz, and Eike Kiltz. Subtleties in the definition of IND-CCA: When and how should challenge-decryption be disallowed? Cryptology ePrint Archive, Report 2009/418, 2009.

[BNN09] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. *Journal of Cryptology*, 22(1):1–61, January 2009.

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer-Verlag, 2002.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracle are practical: A paradigm for designing efficient protocols. In *CCS '93*, pages 62–73. ACM, 1993.

[BR96] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures – how to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 399–416. Springer-Verlag, 1996.

[BS08] Mihir Bellare and Sarah Shoup. Two-tier signatures from the Fiat-Shamir transform, with applications to strongly unforgeable and one-time signatures. *IET Information Security*, 2(2):47–63, 2008.

[Ber08] Daniel J. Bernstein. Proving tight security for Rabin-Williams signatures. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 70–87. Springer-Verlag, 2008.

[BGM93] Ian F. Blake, Shuhong. Gao, and Ronald C. Mullin. Explicit factorization of $x^{2^k} + 1$ over F_p with prime $p \equiv 3 \bmod 4$. *Applicable Algebra in Engineering, Communication and Computing*, 4:89–94, 1993.

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In Kaisa Nyberg, editor, *EUROCRYPT '98*, volume 1403 of *Lecture Notes in Computer Science*, pages 127–144. Springer-Verlag, 1998.

[BB04] Dan Boneh and Xavier Boyen. Secure identity based encrypton without random oracles. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459. Springer-Verlag, 2004.

[BB09] Dan Boneh and Xavier Boyen. Efficient lattice (H)IBE in the standard model from the BB-1 framework. Oral Presentation, August 2009. The slides are available at `http://rump2009.cr.yp.to/`.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):584–615, 2003. Preliminary version in *CRYPTO 2001*, 2001.

[BHHO08] Dan Boneh, Shai Halevi, Mike Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 108–125. Springer-Verlag, 2008.

[BGK09] Zvika Brakerski, Shafi Goldwasser, and Yael Kalai. Circular-secure encryption beyond Affine functions. Cryptology ePrint Archive, Report 2009/485, 2009.

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *Journal of Computer and System Sciences (JCSS)*, 37(2):156–189, October 1988.

[BDS08] Johannes Buchmann, Erik Dahmen, and Michael Szydlo. *Post-Quantum Cryptography*, chapter Hash-based Digital Signature

Schemes, pages 35–93. Springer, Heidelberg, 2008.

[BL09] Johannes Buchmann and Richard Lindner. Density of ideal lattices (extended abstract). WEWoRC 2009, 2009. Available at `http://2009.weworc.org/accepted_talks.html`.

[Cai98a] Jin-Yi Cai. A new transference theorem and applications to Ajtai's connection factor. *Electronic Colloquium on Computational Complexity (ECCC)*, 5(005), 1998.

[Cai98b] Jin-Yi Cai. A relation of primal-dual lattices and the complexity of shortest lattice vector problem. *Theoretical Computer Science*, 207(1):105–116, 1998.

[CN97] Jin-Yi Cai and Ajay Nerurkar. An improved worst-case to average-case connection for lattice problems. In *38th Annual Symposium on Foundations of Computer Science (FOCS '97)*, pages 468–477. IEEE Computer Society, 1997.

[CH07] Ran Canetti and Susan Hohenberger. Chosen-ciphertext secure proxy re-encryption. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *CCS 2007*, pages 185–194. ACM, 2007.

[CC98] Anne Canteaut and Florent Chabaud. A new algorithm for finding minimum-weight words in a linear code: Application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511. *IEEE Transactions on Information Theory*, 44(1):367–378, 1998.

[CHK09] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. Cryptology ePrint Archive, Report 2009/351, 2009. To appear in *EUROCRYPT 2010* as Cash, Hofheinz, Kiltz, and Peikert "Bonsai Trees, or How to Delegate a Lattice Basis.".

[CGDG09] Pierre-Louis Cayrel, Philippe Gaborit, Galindo David, and Marc Girault. Improved identity-based identification using correcting codes. [arXiv:09030069], 2009. Available at `http://arxiv.org/abs/0903.0069`.

[CGG07] Pierre-Louis Cayrel, Philippe Gaborit, and Marc Girault. Identity-based identification and signature scheme using correcting codes. In *WCC 2007*, April 2007.

[CT07] Cheng-Kang Chu and Wen-Guey Tzeng. Identity-based proxy re-encryption without random oracles. In Juan A. Garay, Arjen K. Lenstra, Masahiro Mambo, and René Peralta, editors, *ISC 2007*, volume 4779 of *Lecture Notes in Computer Science*, pages 189–202. Springer-Verlag, 2007.

[Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *IMA 2001*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer-Verlag, 2001.

[CS97] Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In Walter Fumy, editor, *EUROCRYPT '97*, volume 1233 of *Lecture Notes in Computer Science*, pages 52–61. Springer-Verlag, 1997.

[Cor00] Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 229–235. Springer-Verlag, 2000.

[Cor02] Jean-Sébastien Coron. Optimal security proofs for PSS and other signature schemes. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 272–287. Springer-Verlag, 2002.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. How to achieve a McEliece-based digital signature scheme. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 157–174. Springer-Verlag, 2001.

[CS03] Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[Dam89] Ivan Damgård. A design principle for hash functions. In Gilles Brassard, editor, *CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 416–427. Springer-Verlag, 1989.

[DPP97] Ivan B. Damgård, Torben P. Pedersen, and Birgit Pfizmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10(3):163–194, 1997. Preliminary version in *CRYPTO '93*, 1993.

[DPP98] Ivan B. Damgård, Torben P. Pedersen, and Birgit Pfizmann. Statistical secrecy and multibit commitments. *IEEE Transactions on Information Theory*, 44(3):1143–1151, May 1998.

[DSDCPY94] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *35th Annual Symposium on Foundations of Computer Science (FOCS '94)*, pages 454–465. IEEE Computer Society, 1994.

[DWLC08] Robert H. Deng, Jian Weng, Shengli Liu, and Kefei Chen. Chosen-ciphertext secure proxy re-encryption without pairings. In Matthew K. Franklin, Lucas Chi Kwong Hui, and Duncan S.

Wong, editors, *CANS 2008*, volume 5339 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 2008.

[Dev86] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer, Heidelberg, 1986.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, November 1976.

[DL07] Jintai Ding and Richard Lindner. Identifying ideal lattices. Cryptology ePrint Archive, Report 2007/322, 2007.

[DKRS03] Irit Dinur, Guy Kindler, Ran Raz, and Shmuel Safra. Approximating CVP to within almost-polynomial factors is NP-hard. *Combinatorica*, 23(2):205–243, 2003. Preliminary version in *FOCS '98*, 1998.

[DGK$^+$10] Yevgeniy Dodis, Shafi Goldwasser, Yael Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *Lecture Notes in Computer Science*, pages 361–381. Springer-Verlag, 2010.

[DKNS04] Yevgeniy Dodis, Aggelos Kiayias, Antonio Nicolosi, and Victor Shoup. Anonymous identification in *ad hoc* groups. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 609–626. Springer-Verlag, 2004.

[DMQN09] Rafael Dowsley, Jörn Müller-Quade, and Anderson C. A. Nascimento. A CCA2 secure public key encryption scheme based on the McEliece assumptions in the standard model. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *Lecture Notes in Computer Science*, pages 240–251. Springer-Verlag, 2009.

[Eii08] Fujisaki Eiichiro, August 2008. Personal Communications.

[ElG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transaction on Information Theory*, 31(4):469–472, 1985.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings on 22nd Annual ACM Symposium on Theory of Computing (STOC '90)*, pages 416–426. ACM, 1990.

[Fis01] Marc Fischlin. *Trapdoor Commitment Schemes and Their Applications*. PhD thesis, Johann Wolfgang Goethe-Universität, Frankfurt am Main, 2001.

[GG07] Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. In *Information Theory, 2007. ISIT 2007. IEEE International Symposium on*, pages 191–195. IEEE Computer Society, 2007.

[GN07] Nicolas Gama and Phong Q. Nguyen. New chosen-ciphertext attacks on NTRU. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 89–106. Springer-Verlag, 2007.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. Available at `http://crypto.stanford.edu/craig/`.

[GJSS01] Craig Gentry, Jakob Jonsson, Jacques Stern, and Mike Szydlo. Cryptanalysis of the NTRU signature scheme (NSS) from Eurocyrpt 2001. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 1–20. Springer-Verlag, 2001.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *Proceedings on the 40th Annual ACM Symposium on Theory of Computing (STOC 2008)*, pages 197–206. ACM, 2008.

[Gol01] Oded Goldreich. *Foundations of Cryptography: Volume I – Basic Tools*. Cambridge University Press, 2001.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of non-approximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000.

[GGH96] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Collision-free hashing from lattice problems. *Electronic Colloquium on Computational Complexity (ECCC)*, 3(42), 1996.

[GGH97a] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Eliminating decryption errors in the Ajtai-Dwork cryptosystem. In Burton S. Kaliski, Jr., editor, *CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 105–111. Springer-Verlag, 1997. See also ECCC TR97-018.

[GGH97b] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski, Jr., editor, *CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer-Verlag, 1997. See also ECCC TR96-056.

[GL89] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proceedings on 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pages 25–32. ACM, 1989.

[GKPV10] Shafi Goldwasser, Yael Kalai, Chris Peikert, and Vinod Vaikuntanathan. Robustness of the learning with errors assumption. In Andrew Chi-CHih Yao, editor, *ICS 2010*, pages 230–240, Beijing, China, 2010. Tsinghua University Press.

[GV08] Shafi Goldwasser and Vinod Vaikuntanathan. New constructions of correlation-secure trapdoor functions and CCA-secure encryption schemes. Manuscript, 2008.

[GA07] Matthew Green and Giuseppe Ateniese. Identity-based proxy re-encryption. In Jonathan Katz and Moti Yung, editors, *ACNS 2007*, volume 4521 of *Lecture Notes in Computer Science*, pages 288–306. Springer-Verlag, 2007.

[GQ88] Louis Claude. Guillou and Jean-Jacques Quisquater. A "paradoxical" identity-based signature scheme resulting from zero-knowledge. In Shafi Goldwasser, editor, *CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 1988.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment scheme from collision-free hashing. In Neal Koblitz, editor, *CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 201–215. Springer-Verlag, 1996.

[HGS99] Chris Hall, Ian Goldberg, and Bruce Schneier. Reaction attacks against several public-key cryptosystems. In Vijay Varadharajan and Yi Mu, editors, *ICICS '99*, volume 1726 of *Lecture Notes in Computer Science*, pages 2–12. Springer-Verlag, 1999.

[HHHK03] Daewan Han, Jin Hong, Jae Woo Han, and Daesung Kwon. Key recovery attacks on NTRU without ciphertext validation routine. In Reihaneh Safavi-Naini and Jennifer Seberry, editors, *ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 274–284. Springer-Verlag, 2003. See also `http://eprint.iacr.org/2002/188`.

[HKY07] Daewan Han, Myung-Hwan Kim, and Yongjin Yeom. Cryptanalysis of the paeng-jung-ha cryptosystem from pkc 2003. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 107–117. Springer-Verlag, 2007.

[HR07] Ishay Haviv and Oded Regev. Ternsor-based hardness of the shortest vector problem to within almost polynomial factors. In

David S. Johnson and Uriel Feige, editors, *Proceedings on the 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 469–477. ACM, 2007.

[HHK06] Javier Herranz, Dennis Hofheinz, and Eike Kiltz. KEM/DEM: Necessary and sufficient conditions for secure hybrid encryption. Cryptology ePrint Archive, Report 2006/265, 2006.

[HHHGW09] P. Hirschhorn, J. Hoffstein, N. Howgrave-Graham, and W. Whyte. Choosing NTRUEncrypt parameters in light of combined lattice reduction and MITM approaches. In Michel Abdalla, David Pointcheval, Pierre-Alain Fouque, and Damien Vergnaud, editors, *ACNS 2009*, volume 5536 of *Lecture Notes in Computer Science*, pages 437–455. Springer-Verlag, 2009.

[HHGP$^+$03] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. NTRUSign: Digital signature using the NTRU lattice. In Marc Joye, editor, *CT-RSA 2003*, volume 2612 of *Lecture Notes in Computer Science*, pages 122–140. Springer-Verlag, 2003.

[HHGP$^+$07] Jeffrey Hoffstein, Nick Howgrave-Graham, Jill Pipher, Joseph H. Silverman, and William Whyte. Hybrid lattice reduction and meet in the middle resistant parameter selection for NTRU-Encrypt, 2007. Available at `http://grouper.ieee.org/groups/1363/lattPK/submissions.html#2007-02`.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS-III*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer-Verlag, 1998.

[HPS01] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NSS: An NTRU lattice-based signature scheme. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 211–228. Springer-Verlag, 2001.

[HS00] Jeffrey Hoffstein and Joseph Silverman. Optimizations for NTRU, 2000. Available at `http://www.ntru.com/cryptolab/articles.htm`.

[HMS04] Thomas Holenstein, Ueli M. Maurer, and Johan Sjödin. Complete classification of bilinear hard-core functions. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 73–91. Springer-Verlag, 2004.

[HG07] Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against NTRU. In Alfred Menezes, editor, *CRYPTO 2007*, volume 4622 of *Lecture Notes in Computer Sci-*

ence, pages 150–169. Springer-Verlag, 2007.

[HGNP^{+}03] Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer-Verlag, 2003.

[HGSW03] Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. A meet-in-the-middle attack on an NTRU private key. Technical Report 4, version 2, NTRU Cryptosystems, 2003. Available at `http://www.ntru.com/cryptolab/tech_notes.htm`.

[HGSW05] Nick Howgrave-Graham, Joseph H. Silverman, and William Whyte. Choosing parameter sets for NTRUEncrypt with NAEP and SVES-3, 2005. Available at `http://www.ntru.com/cryptolab/articles.htm`.

[IM06] Oleg Izmerly and Tal Mor. Chosen ciphertext attacks on lattice-based public key encryption and modern (non-quantum) cryptography in a quantum environment. *Theoretical Computer Science*, 367(3):308–323, 2006.

[JJ00] Éliane Jaulmes and Antoine Joux. A chosen-ciphertext attack against NTRU. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 20–35. Springer-Verlag, 2000.

[Kat03] Jonathan Katz. Efficient and non-malleable proofs of plaintext knowledge and applications. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 211–228. Springer-Verlag, 2003.

[KK05] Jonathan Katz and Chiu-Yuen Koo. On constructing universal one-way hash functions from arbitrary one-way functions. Cryptology ePrint Archive, Report 2005/328, 2005.

[KL07] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman & Hall/CRC, 2007.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange based on lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 636–652. Springer-Verlag, 2009.

[KY06] Jonathan Katz and Moti Yung. Characterization of security notions for probabilistic private-key encryption. *Journal of Cryp-*

tology, 19(1):67–95, 2006. Preliminary version in *STOC 2000*, 2000.

[KTX07] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Multi-bit cryptosystems based on lattice problems. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 315–329. Springer-Verlag, 2007.

[KTX08] Akinori Kawachi, Keisuke Tanaka, and Keita Xagawa. Concurrently secure identification schemes based on the worst-case hardness of lattice problems. In Josef Pieprzyk, editor, *ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 372–389. Springer-Verlag, 2008.

[KX09] Akinori Kawachi and Keita Xagawa. Simultaneous security of quantum hardcore functions. Manuscript, July 2009.

[KY06] Akinori Kawachi and Tomoyuki Yamakami. Quantum hardcore functions by complexity-theoretical quantum list decoding. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 216–227. Springer-Verlag, 2006. [arXiv:quant-ph/0602088].

[Kho05] Subhash Khot. Hardness of approximating the shortest vector problem in lattices. *Journal of the ACM*, 52(5):789–808, 2005. Preliminary version in *FOCS 2004*, 2004.

[Kho06] Subhash Khot. Hardness of approximating the shortest vector problem in high ℓ_p norms. *Journal of Computer and System Sciences (JCSS)*, 72(2):206–219, 2006. Preliminary version in *FOCS 2003*, 2003.

[Kle00] Philip N. Klein. Finding the closest lattice vector when it's unusually close. In *SODA 2000*, pages 937–941. ACM/SIAM, 2000.

[KR00] Hugo Krawczyk and Tal Rabin. Chameleon signatures. In *NDSS 2000*, pages 143–154. Internet Society, 2000.

[KS01] S.-Ravi Kumar and D. Sivakumar. On the unique shortest lattice vector problem. *Theoretical Computer Science*, 255(1-2):641–648, 2001.

[KH04] Kaoru Kurosawa and Swee-Huay Heng. From digital signature to ID-based identification/signature. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 248–261. Springer-Verlag, 2004.

[KH08] Kaoru Kurosawa and Swee-Huay Heng. The power of identification schemes. *International Journal of Applied Cryptography (IJACT)*, 1(1):60–69, 2008. Preliminary version in *PKC 2006*, 2006.

[Lam79] Leslie Lamport. Constructing digital signatures from a one way function. Technical Report SRI-CSL-98, SRI International Computer Science Laboratory, 1979.

[LH08] Moon Sung Lee and Sang Geun Hahn. Analysis of the GGH cryptosystem. In *Proceedings of First International Conference on Symbolic Computation and Cryptography (SCC 2008)*, Beijing, China, April 2008.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):513–534, 1982.

[LV08] Benoît Libert and Damien Vergnaud. Unidirectional chosen-ciphertext secure proxy re-encryption. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 360–379. Springer-Verlag, 2008.

[Lyu08a] Vadim Lyubashevsky. Lattice-based identification schemes secure under active attacks. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 162–179. Springer-Verlag, 2008.

[Lyu08b] Vadim Lyubashevsky. *Towards Practical Lattice-Based Cryptography*. PhD thesis, University of Calfornia, San Diego, 2008.

[Lyu09] Vadim Lyubashevsky. Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 598–616. Springer-Verlag, 2009.

[LM06] Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 144–155. Springer-Verlag, 2006.

[LM08] Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In Yevgeniy Dodis and Victor Shoup, editors, *TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 37–54. Springer-Verlag, 2008.

[LM09] Vadim Lyubashevsky and Daniele Micciancio. On bounded distance decoding, unique shortest vectors, and the minimum distance problem. In Shai Halevi, editor, *CRYPTO 2009*, volume

5677 of *Lecture Notes in Computer Science*, pages 577–594. Springer-Verlag, 2009.

[LMPR08] Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In Kaisa Nyberg, editor, *FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 54–72. Springer-Verlag, 2008.

[LPS10] Vadim Lyubashevsky, Adriana Palacio, and Gil Segev. Public-key cryptographic primitives provably as secure as subset sum. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *Lecture Notes in Computer Science*, pages 382–400. Springer-Verlag, 2010.

[MNT10] Toshihide Matsuda, Ryo Nishimaki, and Keisuke Tanaka. CCA proxy re-encryption without bilinear maps in the standard model (extended abstract). SCIS 2010, 2010. To appear in *PKC 2010*.

[Mat07] Toshihiko Matsuo. Proxy re-encryption systems for identity-based encryption. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 247–267. Springer-Verlag, 2007.

[May99] Alexander May. Cryptanalysis of NTRU-107, 1999. Available at `http://www.informatik.tu-darmstadt.de/KP/alex.html`.

[McE78] Robert J. McEliece. A public key cryptosystem based on algebraic coding theory. Technical report, DSN progress report, 1978.

[Mer89] Ralph C. Merkle. One way hash functions and DES. In Gilles Brassard, editor, *CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 428–446. Springer-Verlag, 1989.

[MH78] Ralph C. Merkle and Martin E. Hellman. Hiding information and signatures in trap door knapsacks. *IEEE Transactions on Information Theory*, 24(5):525–530, September 1978.

[MR06] Tommi Meskanen and Ari Renvall. A wrap error attack against NTRUEncrypt. *Discrete Applied Mathematics*, 154(2):382–391, 2006. Preliminary version in *WCC 2003*, 2003.

[Mic00] Daniele Micciancio. The shortest vector problem is NP-hard to approximate to within some constant. *SIAM Journal on Computing*, 30(6):2008–2035, 2000.

[Mic01] Daniele Micciancio. Improving lattice based cryptosystems using the Hermite normal form. In Joseph H. Silverman, editor, *CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages

126–145. Springer-Verlag, 2001.

[Mic04] Daniele Micciancio. Almost perfect lattices, the covering radius problem, and applications to Ajtai's connection factor. *SIAM Journal on Computing*, 34(1):118–169, 2004. Preliminary version in *STOC 2002*, 2002.

[Mic07] Daniele Micciancio. Generalized compact knapsacks, cyclic lattices, and efficient one-way functions. *Computational Complexity*, 16:365–411, 2007. Preliminary version in *FOCS 2002*, 2002. See also ECCC TR04-095.

[MG02] Daniele Micciancio and Shafi Goldwasser. *Complexity of Lattice Problems: a cryptographic perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, Massachusetts, March 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. Preliminary version in *FOCS 2004*, 2004.

[MR08] Daniele Micciancio and Oded Regev. *Post-Quantum Cryptography*, chapter Lattice-based Cryptography, pages 147–191. Springer, Heidelberg, 2008.

[MV03] Daniele Micciancio and Salil Vadhan. Statistical zero-knowledge proofs with efficient provers: Lattice problems and more. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 282–298. Springer-Verlag, 2003.

[MV09] Daniele Micciancio and Panagiotis Voulgaris. Faster exponential time algorithms for the shortest vector problem. *Electronic Colloquium on Computational Complexity (ECCC)*, (065), 2009. To appear in *SODA 2010*.

[MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. *Electronic Colloquium on Computational Complexity (ECCC)*, (014), 2010. To appear in *STOC 2010*.

[MY08] Petros Mol and Moti Yung. Recovering NTRU secret key from inversion oracles. In Ronald Cramer, editor, *PKC 2008*, volume 4939 of *Lecture Notes in Computer Science*, pages 18–36. Springer-Verlag, 2008.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer-Verlag, 2009.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings on 21st Annual ACM Symposium on Theory of Computing (STOC '89)*, pages 33–43. ACM, 1989.

[NSW03] Mats Näslund, Igor E. Shparlinski, and William Whyte. On the bit security of NTRUEncrypt. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 62–70. Springer-Verlag, 2003.

[Ngu99] Phong Q. Nguyen. Cryptanalysis of the Goldreich-Goldwasser-Halevi cryptosystem from Crypto '97. In Michael J. Wiener, editor, *CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 288–304. Springer-Verlag, 1999.

[NR06] Phong Q. Nguyen and Oded Regev. Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 271–288. Springer-Verlag, 2006.

[NS98] Phong Q. Nguyen and Jacques Stern. Cryptanalysis of the Ajtai-Dwork cryptosystem. In Hugo Krawczyk, editor, *CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 223–242. Springer-Verlag, 1998.

[NS01] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In Joseph H. Silverman, editor, *CaLC 2001*, volume 2146 of *Lecture Notes in Computer Science*, pages 146–180. Springer-Verlag, 2001.

[NV08] Phong Q. Nguyen and Thomas Vidick. Sieve algorithms for the shortest vector problem are practical. *Journal of Mathematical Cryptology*, 2(2):181–207, 2008.

[OO98] Kazuo Ohta and Tatsuaki Okamoto. On concrete security treatment of signatures derived from identification. In Hugo Krawczyk, editor, *CRYPTO '98*, volume 1462 of *Lecture Notes in Computer Science*, pages 354–369. Springer-Verlag, 1998.

[OS08] Raphael Overbeck and Nicolas Sendrier. *Post-Quantum Cryptography*, chapter Code-based Cryptography, pages 95–145. Springer, Heidelberg, 2008.

[PJH03] Seong-Hun Paeng, Bae Eun Jung, and Kil-Chan Ha. A lattice based public key cryptosystem using polylnomial representations. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *Lecture Notes in Computer Science*, pages 292–308. Springer-

Verlag, 2003.

[Pei07] Chris Peikert. Limits on the hardness of lattice problems in l_p norms. In *CCC 2007*, pages 333–346. IEEE Computer Society, 2007. See also ECCC TR06-148.

[Pei08] Chris Peikert. Limits on the hardness of lattice problems in l_p norms. *Computational Complexity*, 17(2):300–351, May 2008. Preliminary version in *CCC 2007*, 2007.

[Pei09a] Chris Peikert, February 2009. Personal Communications.

[Pei09b] Chris Peikert. Bonsai trees (or, arboriculture in lattice-based cryptography). Cryptology ePrint Archive: Report 2009/359, 2009. To appear in *EUROCRYPT 2010* as Cash, Hofheinz, Kiltz, and Peikert "Bonsai Trees, or How to Delegate a Lattice Basis.".

[Pei09c] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *Proceedings on the 41st Annual ACM Symposium on Theory of Computing (STOC 2009)*. ACM, 2009.

[PR06] Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In Shai Halevi and Tal Rabin, editors, *TCC 2006*, volume 3876 of *Lecture Notes in Computer Science*, pages 145–166. Springer-Verlag, 2006.

[PR07] Chris Peikert and Alon Rosen. Lattices that admit logarithmic worst-case to average-case connection factors. In David S. Johnson and Uriel Feige, editors, *Proceedings on the 39th Annual ACM Symposium on Theory of Computing (STOC 2007)*, pages 478–487. ACM, 2007.

[PV08] Chris Peikert and Vinod Vaikuntanathan. Non-interactive statistical zero-knowledge for lattice problems. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 536–553. Springer-Verlag, 2008.

[PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO 2008*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer-Verlag, 2008.

[PW08] Chris Peikert and Brent Waters. Lossy trapdoor functions and their applications. In Richard E. Ladner and Cynthia Dwork, editors, *Proceedings on the 40th Annual ACM Symposium on Theory of Computing (STOC 2008)*, pages 187–196. ACM, 2008.

[PRS09] Thomas Plantard, Michael Rose, and Willy Susilo. Improvement of lattice-based cryptography using CRT. In Alexander

Sergienko, Saverio Pascazio, and Paolo Villoresi, editors, *QuantumComm 2009*, volume 36 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 275–282. Springer-Verlag, 2009.

[PP03] David Pointcheval and Guillaume Poupard. A new NP-complete problem and public-key identification. *Designs, Codes and Cryptography*, 28(1):5–31, January 2003.

[PS96] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer-Verlag, 1996.

[Reg03] Oded Regev. New lattice based cryptographic constructions. In *Proceedings on the 35th Annual ACM Symposium on Theory of Computing (STOC 2003)*, pages 407–416. ACM, 2003.

[Reg04a] Oded Regev. Lattices in computer science. Lecture Notes, 2004. Available at `http://www.cs.tau.ac.il/~odedr/teaching/lattices_fall_2004/`.

[Reg04b] Oded Regev. New lattice-based cryptographic constructions. *Journal of the ACM*, 51(6):899–942, 2004. Preliminary version in *STOC 2003*, 2003.

[Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *Journal of the ACM*, 56(6):Article 34, 2009. Preliminary version in *STOC 2005*, 2005.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, Febrary 1978.

[RS04] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *FSE 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer-Verlag, 2004.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings on 22nd Annual ACM Symposium on Theory of Computing (STOC '90)*, pages 387–394. ACM, 1990.

[RS09] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. In Omer Reingold, editor, *TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 419–436.

Springer-Verlag, 2009.

[Rot06] Ron M. Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006.

[Rüc10] Markus Rückert. Strongly unforgeable signatures and hierarchical identity-based signatures from lattices without random oracles. Cryptology ePrint Archive, Report 2010/070, 2010. To appear in *PQCrypto 2010*.

[SOK01] Ryuichi Sakai, Kiyoshi Ohgishi, and Masao Kasahara. Cryptosystems based on pairing over elliptic curve (in japanese). In *SCIS 2001*, Oiso, Japan, January 2001.

[Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theoretical Computer Science*, 53(2-3):201–224, 1987.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *CRYPTO '84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985.

[Sha89] Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract). In Gilles Brassard, editor, *CRYPTO '89*, volume 435 of *Lecture Notes in Computer Science*, pages 606–609. Springer-Verlag, 1989.

[ST01] Adi Shamir and Yael Tauman. Improved online/offline signature schemes. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 355–367. Springer-Verlag, 2001.

[Sho97] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[Sho08] Victor Shoup. *A Computational Introduction to Number Theory and Algebra*. Cambridge University Press, second edition, 2008.

[SV09] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. Cryptology ePrint Archive, Report 2009/571, 2009. To appear in *PKC 2010*, 2010.

[SCM01] Patrick Solé, Chris Charnes, and Bruno Martin. A lattice-based McEliece scheme for encryption and signature. *Electronic Notes in Discrete Mathematics*, 6:402–411, April 2001. The Proceedings of International Workshop on Coding and Cryptography 2001.

[SS09] Damien Stehlé and Ron Steinfeld, August 2009. Personal Communications.

[SSTX09] Damien Stehlé, Ron Steinfeld, Keisuke Tanaka, and Keita Xagawa. Efficient public key encryption based on ideal lattices. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 617–635. Springer-Verlag, 2009.

[Ste96] Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory*, 42(6):749–765, November 1996. Preliminary version in *CRYPTO '93*, 1993.

[vDGHV09] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. Cryptology ePrint Archive, Report 2009/615, 2009.

[vEB81] Peter van Emde Boas. Another NP-complete partition problem and the complexity of computing short vectors in a lattice. Technical Report MI-UvA-81-04, Mathematische Instituut, University of Amsterdam, 1981. Available at `http://staff.science.uva.nl/~peter/vectors/mi8104c.html`.

[vN51] John von Neumann. Various techniques used in connection with random digits. Monte Carlo methods. *Nat. Bureau Standards*, 12:36–38, 1951.

[WHGH$^+$08] William Whyte, Nick Howgrave-Graham, Jeff Hoffstein, Jill Pipher, Joseph H. Silverman, and Phil Hirschhorn. IEEE P1363.1/D12 draft standard for public-key cryptographic techniques based on hard problems over lattices, October 2008. Available at `http://grouper.ieee.org/groups/1363/lattPK/`.

[XT09] Keita Xagawa and Keisuke Tanaka. Zero-knowledge protocols for NTRU: Application to identification and proof of plaintext knowledge. In Josef Pieprzyk and Fangguo Zhang, editors, *ProvSec 2009*, volume 5848 of *Lecture Notes in Computer Science*, pages 198–213. Springer-Verlag, 2009.

[YCW$^+$07] Guomin Yang, Jing Chen, Duncan S. Wong, Xiaotie Deng, and Dongsheng Wang. A more natural way to construct identity-based identification schemes. In Jonathan Katz and Moti Yung, editors, *ACNS 2007*, volume 4521 of *Lecture Notes in Computer Science*, pages 307–322. Springer-Verlag, 2007.