SCloud

Public Key Encryption and Key Encapsulation Mechanism
Based on Learning with Errors

Zhongxiang Zheng[1], Anyu Wang[1], Haining Fan[2],
Chunhuan Zhao[1], Chao Liu[3], and Xue Zhang[1]

zhengzx13@mails.tsinghua.edu.cn

[1]Institute for Advanced Study, Tsinghua University, Beijing 100084, China

[2]Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

[3]Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan 250100, China

Contents

Abstract

We propose a new family of public key encryption (PKE) and key encapsulation mechanism (KEM) schemes based on the plain learning with errors (LWE) problem. Two new design techniques are adopted in the proposed scheme named SCloud: the sampling method and the error-reconciliation mechanism. The new sampling method is obtained by studying the property of the convolution of central binomial distribution and bounded uniform distribution which can achieve higher efficiency and more flexibility w.r.t the parameter choice. Besides, it is shown to be more secure against the dual attack due to its advantage in distinguish property. The new error-reconciliation mechanism is constructed by combining the binary linear codes and Gray codes. It can reduce the size of parameters, and then improve the encryption/decryption efficiency as well as communication efficiency, by making full use of the encryption space. Based on these two techniques, SCloud can provide various sets of parameters for refined security level.

1 Introduction and Design Rationale

We propose a new family of public key encryption (PKE) and key encapsulation mechanism (KEM) schemes based on the *plain* learning with errors (LWE) problem. The PKE schemes include SCloudCPAPKE and SCloudCCAPKE, which achieve IND-CPA security and IND-CCA security respectively, and the KEM scheme SCloudKEM achieves the IND-CCA security. Their relationship is as follows. SCloudKEM is obtained by applying the Fujisaki-Okamoto transformation to SCloudCPAPKE, and SCloudCCAPKE is constructed by combining SCloudKEM and a data-encapsulate mechanism.

1.1 Design Rationale

Many current public-key post-quantum cryptosystems are based on the hardness of the LWE problem. Simply speaking, the LWE problem is to solve a linear equation system, with some of the equations are erroneous, modulo a known positive integer. The LWE problem is closely related to the computational lattice problems. In 2005, Regev proves the hardness of the LWE problem by assuming the quantum hardness of the Shortest Independent Vectors Problem (SIVP) on random lattices. In 2009, Peikert presents a similar result assuming only the classical hardness of another lattice problem. Besides, it is proved that the LWE problem can be transformed into solving the Shortest Vector Problem (SVP) on a related lattice.

Unstructured Lattices v.s. Structured Lattices. For the sake of computational efficiency, some variants of the LWE problem, e.g., RLWE [23, 26], MLWE [9, 21] and NTRU [29], are adopted in the construction of public-key cryptosystems. Different to the random unstructured lattices related to the *plain* LWE problem, these variant LWE problems are proved to be as hard as the problems on lattices with algebraic structures [23, 21, 26], such as ideal lattices or module lattices.

Though there is *no* variant LWE based cryptosystems can be cracked under recommended parameters now, it is still risky to use structured lattices from the security perspective. In [19, 20], an attack against NTRU can significantly reduce the asymptotic complexity compared with that estimated by ignoring the structure of related lattices. Recently, a polynomial-time quantum algorithm is proposed for SVP with approximation factor $2^{\tilde{O}(\sqrt{n})}$ over a widely used class of rings [12], while the best known algorithm can only achieves approximation factor $2^{O(n \log\log n/\log n)}$ for unstructured lattices [28].

As a result, the design of SCloud is based on the *plain* LWE problem (related to unstructured lattices), rather than based on the variant LWE problems, such as RLWE, MLWE, etc., (related to structured lattices). Though at a slight expense of performance (in terms of public and private key size and operation efficiency), taking into consideration the future development of quantum computation, the design of SCloud aims to provide a long term security against various attacks in the future. Moreover, benefit from the new techniques adopted in our design, SCloud is still efficient in most practical environment.

Our Technique. Two new design techniques are adopted in SCloud: the sampling method and the error-reconciliation mechanism. The new sampling method is obtained by studying the property of the convolution of central binomial distribution and bounded uniform distribution. It can achieve higher efficiency and more flexibility w.r.t the parameter choice. Besides, it is shown to be more secure against the dual attack due to its advantage in distinguish property [11]. The new error-reconciliation mechanism is constructed by combining the binary linear codes and Gray codes. It can reduce the size of parameters, and then improve the encryption/decryption efficiency as well as communication efficiency, by making full use of the encryption space.

Based on these two techniques, SCloud can provide various sets of parameters for refined security level.

- High efficiency & Long message mode:
  SCloud-f640, SCloud-f896 and SCloud-f1216 (see Table 1).
- High efficiency & Short message mode:
  SCloud-fg640, SCloud-fg896 and SCloud-fg1216 (see Table 2).
- High security & Long message mode:
  SCloud-q640, SCloud-q896 and SCloud-q1216 (see Table 3).
- High security & Short message mode:
  SCloud-qg640, SCloud-qg896 and SCloud-qg1216 (see Table 4).

High security mode provides parameters that satisfy Regev's reduction condition [27] so as to keep security against known attacks including primal attack, dual attack, BKW and algebra attack.

High efficiency mode provides better performance and smaller size with practical security against primal attack and dual attack which are the most common methods considered in lattice-based designs.

Long message mode refers to the message length is the double security level (in bits), which takes into consideration the quantum acceleration of the exhaustive search of message.

Short message mode refers to the message length is equal to the security level (in bits).

Roadmap In Section 2, we give some basic notations and backgrounds. The algorithm specification is presented in Section 3, while the sets of parameters are provided in Section 4. In Section 5 and Section 6, we analysis the security and performance respectively. Section 7 concludes the advantages and the limitations of the proposed schemes.

2 Preliminary

2.1 Notations

We will use the following notations:

- Let bold lower-case letters denote vectors (e.g. $\mathbf{a}, \mathbf{b}, \mathbf{v}$), the i-th entry of an n-dimensional vector $\mathbf{v}$ is denoted by v_i $(0 \leq i < n)$.
- Let bold upper-case letters denote matrices (e.g. $\mathbf{A}, \mathbf{B}$), the (i, j)-th entry of an $m \times n$ matrix $\mathbf{A}$ is denoted by $\mathbf{A}_{i,j}$ $(0 \leq i < m$ and $0 \leq j < n)$ and the ith row is denoted by $\mathbf{A}_i = (\mathbf{A}_{i,0}, \mathbf{A}_{i,1}, \cdots, \mathbf{A}_{i,n-1})$.
- For a set D, the set of m-dimensional vectors with entries in D is denoted by D^m, and the set of $m \times n$ matrices with entries in D is denoted by $D^{m \times n}$.
- Let $\mathbb{Z}$ denote the ring of integers, and let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the quotient ring of integers modulo a positive integer q.

- Given a distribution χ, let the $e \leftarrow \chi$ denote sampling a value e from χ. Let χ^n denote the n-fold product distribution of χ with itself.
- Let $U(S)$ denote the uniform distribution on a finite set S.
- Let $\lfloor a \rfloor$ denote the largest integer less than or equal to a, and let $\lfloor a \rceil = \lfloor a + 1/2 \rfloor$ denote the closest integer to a.
- For a real vector $\mathbf{v} \in \mathbb{R}^n$, let $\|\mathbf{v}\|$ denote its Euclidean norm.
- For two n-dimensional vectors $\mathbf{a}, \mathbf{b}$, let $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{n-1} a_i \cdot b_i$ denote their inner product.
- Let $w_H(\boldsymbol{v})$ denote the Hamming weight of a vector $\boldsymbol{v}$.

2.2 Background

2.2.1 Lattices

An n-dimensional full rank lattice $\mathcal{L}$ is a discrete additive subgroup in $\mathbb{R}^n$, such that $span_{\mathbb{R}}(\mathcal{L}) = \mathbb{R}^n$. The vectors in $\mathcal{L}$ can be represented as integer combinations of a basis $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n\}$, i.e.,

$$\mathcal{L} = \mathcal{L}(\mathbf{B}) := \mathbf{B} \cdot \mathbb{Z}^n = \{\sum_{i=1}^{n} z_i \cdot \mathbf{b}_i : z_i \in \mathbb{Z}\},$$

where the vectors in $\mathbf{B}$ are linearly independent over $\mathbb{R}$.

2.2.2 Gaussian Distribution over Lattices

Definition 1 (Gaussian Function). *For a real number $s > 0$, the Gaussian function is defined as*

$$\rho_s(x) := \exp\left(-\pi \|\boldsymbol{x}\|^2 / s^2\right)$$

for $\boldsymbol{x} \in \mathbb{R}^m$ where s is called the width.

Definition 2 (Gaussian Distribution). *For a real number $s > 0$, the Gaussian distribution is denoted as D_s and its probability density function is $D_s(x) = \rho_s(x)/s$ where the standard deviation is $\sigma = s/\sqrt{2\pi}$。*

Definition 3 (Discrete Gaussian Distribution). *p For lattice $\mathcal{L}$ and width s, the discrete Gaussian distribution $D_{s,\mathcal{L}}$ is defined as $D_{s,\mathcal{L}} = \frac{\rho_s(\mathbf{x})}{\rho_s(\mathcal{L})}$ where $\mathbf{x} \in \mathcal{L}$ and $\rho_s(\mathcal{L}) = \sum_{\mathbf{v}\in\mathcal{L}} \rho_s(\mathbf{v})$.*

2.2.3 The LWE Problem

Definition 4 (LWE Distribution). *Let n, q be positive integers, and let χ be a distribution on $\mathbb{Z}$. Given $\mathbf{s} \in \mathbb{Z}_q^n$, choosing $\mathbf{a} \leftarrow U(\mathbb{Z}_q^n)$ and $e \leftarrow \chi$, the* LWE *distribution $\mathcal{A}_{s,\chi}$ outputs $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s}\rangle + e \ mod \ q) \in \mathbb{Z}_q^n \times \mathbb{Z}$.*

There are two versions of the LWE problem, i.e., the search version and the decision version. For the two versions of the LWE problem, the distribution of $\mathbf{s} \in \mathbb{Z}_q^n$ can be considered as uniform (called uniform secret) or $\chi^n \mod q$ (called normal form secret).

Definition 5 (Search-LWE). *Let n, m, q be positive integers and let χ be a distribution on $\mathbb{Z}$. The uniform-secret (normal-form-secret) search-LWE with parameters (n, m, q, χ) (called $SLWE_{n,m,q,\chi}$ or $nf\text{-}SLWE_{n,m,q,\chi}$) is that: given m LWE samples with a fixed secret $\mathbf{s} \in \mathbb{Z}_q^n$, find $\mathbf{s}$.*

Definition 6 (Decision-LWE). *Let n, m, q be positive integers and let χ be a distribution on $\mathbb{Z}$. The uniform-secret (normal-form-secret) decision-LWE with parameters (n, m, q, χ) (called $DLWE_{n,m,q,\chi}$ or $nf\text{-}DLWE_{n,m,q,\chi}$) is that: given m samples chosen form LWE distribution with a fixed secret $\mathbf{s} \in \mathbb{Z}_q^n$ or uniform distribution, decide which distribution the samples follow.*

2.3 PKE and KEM

Definition 7 (Public Key Encryption). *A public key encryption scheme* PKE *is a tuple of probabilistic polynomial-time (PPT) algorithms (KeyGen, Encrypt,* *Decrypt) along with a message space M:*

- *KeyGen(1^κ): The key generation algorithm takes as input the security parameter 1^κ, then outputs a public/secret key pair (pk, sk).*
- *Encrypt($pk, \mathbf{m}$): The encryption algorithm takes as input a public key pk and a message $\mathbf{m} \in \mathcal{M}$ and outputs a corresponding ciphertext C.*
- *Decrypt(sk, C):The decryption algorithm takes as input a secret key sk and a ciphertext C and outputs a message* **m** *or '$\perp$' (decryption failure).*

Definition 8 (δ-correctness for PKEs [18])**.** *A* PKE *shceme with message space $\mathcal{M}$ is said to be δ-correct if*

$$\mathbb{E}\left[\underset{\mathbf{m}\in\mathcal{M}}{max}\, Pr[\text{PKE.Dec}(sk, c) \neq \mathbf{m} : c \leftarrow \text{PKE.Enc}(pk, \mathbf{m})]\right] \leq \delta,$$

*where the expectation is taken over (pk, sk) ←*PKE.KeyGen()*.*

Definition 9 (Key Encapsulation Mechanism)**.** *A key encapsulation mechanism* (KEM) *is a tuple of PPT algorithms (KeyGen, Encaps, Decaps):*

- *KeyGen(1^κ): The key generation algorithm takes as input the security parameter 1^κ, then it outputs a public/secret key pair (pk, sk).*
- *Encaps(pk)*: *The encapsulation algorithm takes as input a public key pk and outputs a ciphertext C and an encapsulated key $K \in \mathcal{K}$.*
- *Decaps(sk, C): The decapsulation algorithm takes as input a secret key sk and a ciphertext C, then it outputs a key K.*

Definition 10 (δ-correctness for KEMs [18])**.** *A* KEM *shceme is said to be δ-correct if*

$$\Pr\left[Decaps(sk, c) \neq K : (pk, sk) \leftarrow KeyGen(1^\kappa); (c, K) \leftarrow Encaps(pk)\right] \leq \delta.$$

Definition 11 (IND-CPA Security of PKE)**.** *Let* PKE= (KeyGen, Encrypt, Decrypt) *be a public key encryption scheme. Let us define the following experiment between an adversary $\mathcal{A}$ and a challenger:*

Experiment IND-CPA$^b_{PKE,\mathcal{A}}(k)$:

- *The challenger runs $(pk, sk) \leftarrow KeyGen(params)$, and gives pk to $\mathcal{A}$;*
- *The adversary $\mathcal{A}$ outputs two message $(\mathbf{m}_0, \mathbf{m}_1)$ of the same length;*
- *The challenger computes Encrypt$(pk, \mathbf{m}_b)$ and gives it to $\mathcal{A}$;*
- *$\mathcal{A}$ outputs a bit b'; The challenger returns b' as the output of the game.*

The advantage of the adversary $\mathcal{A}$ for breaking the IND-CPA security of a PKE *is defined as*

$$\mathbf{Adv}^{IND-CPA}_{PKE,\mathcal{A}} = |Pr[IND-CPA^1_{PKE,\mathcal{A}}(k) = 1] - Pr[IND-CPA^0_{PKE,\mathcal{A}}(k) = 1]|$$

then for any polynomial time adversary $\mathcal{A}$ and any k, if ϵ is a negligible function, condition $\mathbf{Adv}^{IND-CPA}_{PKE,\mathcal{A}} \leq \epsilon(k)$ holds, we say that PKE *is IND-CPA secure.*

Definition 12 (IND-CCA Security of KEM)**.** *Let* KEM=(KeyGen, Encaps, Decaps) *be a key encapsulation mechanism. Let us define the following experiment between an adversary $\mathcal{A}$ and a challenger:*

Experiment IND-CCA$^b_{KEM,\mathcal{A}}(k)$:

- *The challenger runs $(pk, sk) \leftarrow KeyGen(params)$, and gives pk to $\mathcal{A}$;*
- *$\mathcal{A}$ queries to the decapsulation oracle Decaps$(sk, \cdot)$;*
- *The challenger computes $(c^*, K^*_0) \leftarrow$ Encaps(pk) and $K^*_1 \leftarrow \mathcal{K}$, and the challenger gives (c^*, K^*_b) to $\mathcal{A}$;*

- *$\mathcal{A}$ continues to query the decapsulation oracle, but may not be allowed to query the ciphertext c^*. At the end, $\mathcal{A}$ outputs a bit b'. The challenger returns b' as the output of the game.*

The advantage of $\mathcal{A}$ for breaking the IND-CCA security of KEM *is defined as*

$$\mathbf{Adv}_{KEM,\mathcal{A}}^{IND-CCA} = |Pr[IND-CCA_{KEM,\mathcal{A}}^{1}(k) = 1] - Pr[IND-CCA_{KEM,\mathcal{A}}^{0}(k) = 1]|$$

then for any polynomial time adversary $\mathcal{A}$ and any k, if ϵ is a negligible function, condition $\mathbf{Adv}_{KEM,\mathcal{A}}^{IND-CCA} \leq \epsilon(k)$ holds, we say that KEM *is IND-CCpA secure.*

3 Algorithm Description

To begin with, we introduce the two new techniques adopted in our schemes, i.e., the sampler in Section 3.1 and the error-reconciliation method in Section 3.2.

3.1 A New Sampler

In a LWE-based cryptosystem, the error sampler works as a basis module which not only influences the efficiency of the whole scheme, but also directly affects the security against known attacks. In practice, three types of error samplers are commonly used in lattice-based schemes, namely approximate discrete Gaussian sampler, central binomial sampler and bounded uniform sampler. As for approximate discrete Gaussian sampler and central binomial sampler, a random number generator is needed. Generally, the efficiency of random number generator(i.e. the ratio between bits and seconds) is stable. Therefore, the efficiency of sampling algorithm is determined by the utilization of random number (i.e. the ratio between bits and samples) that outputted by the random number generator. However, random sources are not needed for bounded uniform sampler. The error following bounded uniform distribution is generated by taking a $x \in \mathbb{Z}_q$ as input and calculating $\lfloor \frac{xq}{p} \rceil (p < q)$. Therefore, the output of bounded uniform sampler is deterministic and it follows the uniform distribution over $[-\frac{q}{2p}, \frac{q}{2p}]$ under the random assumption.

In this section, we propose a new sampler denoted as "mixed sampling ". The distribution that the proposed sampler follows is denoted as "mixed distribution". The mixed distribution is generated by the convolution of variables that follow central binomial distribution and variables that follow bounded uniform distribution. We firstly present the definition.

Definition 13 (mixed distribution)**.** *Let k_1, k_2 be positive integers, $\{X_1, \cdots, X_{k_1}\}$ is a sequence of independent and identically distributed variables where $Pr[\mathbf{X}_i = 0] = 1/2, Pr[\mathbf{X}_i = 1] = Pr[\mathbf{X}_i = -1] = 1/4$, $\{Y_1, \cdots, Y_{k_2}\}$ is a sequence of independent and identically distributed variables where $Y_i \sim U[-1, 0, 1]$, the variable X following "mixed distribution" denoted as $\chi(k_1, k_2)$ is the convolution of $\{X_i\}_{i=1}^{k_1}$ and $\{Y_i\}_{i=1}^{k_2}$, i.e.*

$$X = X_1 + \cdots + X_{k_1} + Y_1 + \cdots + Y_{k_2}.$$

Lemma 1. *Let $X \sim \chi(k_1, k_2)$, then*

$$E[X] = 0, D[X] = \frac{k_1}{2} + \frac{2k_2}{3}.$$

Proof. Since $X \sim \chi(k_1, k_2)$, then it could be express of the following form,

$$X = \sum_{i=1}^{k_1} (b_i - b_i') + \sum_{i=1}^{k_2} u_i.$$

where $b_i, b_i' \sim U[0, 1]$ and $Pr[b = 0] = Pr[b = 1] = 1/2$, $u_i \sim U[-1, 0, 1]$ and $Pr[u_i = j] = 1/3$ for $j = -1, 0, 1$. Therefore we have

$$E[X] = 2k_1 E[b_i] + k_2 E[u_i] = 0,$$
$$D[X] = 2k_1 D[b_i] + k_2 D[u_i] = \frac{k_1}{2} + \frac{2k_2}{3}.$$

□

Mixed sampler outputs a convolution distribution of central binomial distributions and bounded uniform distributions where more flexible choices in sampling widths are allowed, compared to that for the central binomial sampler. Furthermore, by choosing parameters properly, the mixed sampler can also achieve better efficiency and security compared with former samplers. We make a brief comparison between mixed sampler and former samplers concerning security, efficiency and parameter choice respectively.

- **Security.** As for LWE parameters, primal attack and dual attack are two mainly methods in estimating the concrete hardness. The length of error distribution is concerned in the calculation of primal attack while the width of error distribution is closely related to the whole complexity of parameters under dual attack. In the dual attack, the error distribution is usually analyzed as the ideal discrete Gaussian distribution. Overall, the current analysis uses width parameters of error distributions to measure the security of error samplers and no attack that deals with the structures of error distributions has been ever considered. Recently, [11] studies the distributions for sampling errors by means of Fourier analysis. Instead of dealing with the error distribution as ideal discrete Gaussian distribution, it directly calculates the distinguish advantage corresponding to concrete error distribution and provides a measure model to describe the difference between practical distributions and ideal Gaussian distribution under dual attack. The results show that the approximate discrete rounded Gaussian sampler and the bounded uniform sampler have gaps compared with the ideal Gaussian sampler while the central binomial sampler shares the same property with ideal one under the measure model. In order to ensure security, the mixed distribution in our scheme also shares the same property with ideal discrete Gaussian distribution, that is, the lower bound of distinguish advantage is the current estimation $\epsilon = e^{-\pi s^2 l^2 / q^2}$ and the complexity under dual attack of LWE parameters with error following mixed distribution is conservative. We give a brief description of the distinguish property of error distributions and mixed sampling algorithm in Appendix A and B. More detailed descriptions and proofs can be found in [11].

- **Efficiency.** The comparison of sampling efficiency of mixed sampling and approximate discrete Gaussian sampling, central binomial distribution and bounded uniform sampling is shown in Table 14. As is shown in Figure 7, mixed sampling with proper parameter choice behaves better when compared with approximate discrete Gaussian sampling and central binomial distribution.

- **Parameter Choice.** Two parameters k_1, k_2 are used in determining mixed distribution while there is only one parameter in central binomial distribution. Therefore, mixed distribution has a wider parameter choice when compared central binomial distribution. Since the width of error distribution is closely related to the failure probability of the scheme and there are mutually limitations among the width and parameters n, q in order to ensure the security of scheme, a wider range of parameter selection in error distribution will make the overall parameters of the scheme more flexible and improve the efficiency of the scheme.

3.2 Error-Reconciliation Mechanism

We design a new error-reconciliation method based on binary linear codes and Gray codes.

Definition 14 (Binary Linear Codes). *An $[n_e, l_e, d_e]$ binary linear code $\mathcal{C}$ is an l_e dimensional linear subspace of $\mathbb{Z}_2^{n_e}$, such that the hamming distance between any two distinct vectors is at least d_e. The decoding radius of $\mathcal{C}$ is defined to be $t_e = \lceil \frac{d_e - 1}{2} \rceil$.*

There are two algorithms associated with the binary linear code $\mathcal{C}$, i.e., the encoding algorithm *BCE($\cdot$)* and the decoding algorithm *BCD($\cdot$)*.

- *BCE($\boldsymbol{u}$)*: input a binary vector $\boldsymbol{u} \in \{0,1\}^{l_e}$, output a binary vector $\boldsymbol{c} \in \{0,1\}^{n_e}$.

- *BCD($\boldsymbol{c}$)*: input a binary vector $\boldsymbol{c} \in \{0,1\}^{n_e}$. If there exists $\boldsymbol{u} \in \{0,1\}^{l_e}$ such that $w_H(BCE(\boldsymbol{u}) - \boldsymbol{c}) \leq t_e$, then output $\boldsymbol{u}$, else a decoding failure occurs and output '$\perp$' (decoding failure).

Definition 15 (Gray Codes)**.** *An h-bit Gray code is an one-to-one map from the binary vectors $\{0,1\}^h$ to the set of integers $\mathbb{Z}_{2^h}$, such that the pre-image of any two adjacent integers differ in only* 1 *bits.*

We denote by *GrayE($\cdot$)* the map from $\{0,1\}^h$ to $\mathbb{Z}_{2^h}$, and denote *GrayD($\cdot$)* its inverse, i.e.,

- *GrayE($\boldsymbol{v}$)*: input a binary vector $\boldsymbol{v} \in \{0,1\}^h$, output an integer $x \in \mathbb{Z}_{2^h}$.
- *GrayD(x)*: input an integer $x \in \mathbb{Z}_{2^h}$, output a binary vector $v \in \{0,1\}^h$. For any $x, x' \in \mathbb{Z}_{2^h}$, we have $|x - x'| = 1 \mod 2^h \Rightarrow w_H(\mathit{GrayD}(x) - \mathit{GrayD}(x')) = 1$.

3.2.1 Message Encoder and Decoder

Suppose $n_e = h \times \bar{m} \times \bar{n}$ and $2^h \mid q$, then a function *MsgEnc($\cdot$)* is employed to encode the binary message $\boldsymbol{m} \in \{0,1\}^{l_e}$ to a matrix $\mathbf{K} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$. The *MsgEnc($\cdot$)* function, which is shown in Algorithm 1, consists of three steps. Firstly, the message $\boldsymbol{m}$ is mapped to a binary vector $\boldsymbol{u}$ of length n_e by the encoding function *BCE($\cdot$)* of an $[n_e, l_e, d_e]$ binary linear code. Secondly, the binary vector $\boldsymbol{u}$ is mapped to an $\bar{m} \times \bar{n}$ matrix $\mathbf{W}$ over $\mathbb{Z}_{2^h}$ through the Gray code, i.e.,

$$\mathbf{W}_{i,j} = \mathit{GrayE}(\mathbf{u}^{(i,j)}), \forall 0 \leq i < \bar{m}, 0 \leq j < \bar{n},$$

where $\mathbf{u}^{(i,j)} = (\mathbf{u}_{(i+\bar{m}j)h}, \mathbf{u}_{(i+\bar{m}j)h+1}, \cdots, \mathbf{u}_{(i+\bar{m}j)h+h-1})$. Thirdly, the matrix $\mathbf{K} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$ is obtained by multiplying each position of $\mathbf{W}$ by $\frac{q}{2^h}$.

On the other hand, the function *MsgDec($\cdot$)*, which is shown in Algorithm 2, decodes the matrix $\mathbf{K} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$ to the binary message $\boldsymbol{m} \in \{0,1\}^{l_e}$ by reversely

Algorithm 1 MsgEnc

Input: Message $\boldsymbol{m} \in \{0,1\}^{l_e}$

Output: Matrix $\mathbf{K} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$.

```
1: $\boldsymbol{u} \leftarrow BCE(\boldsymbol{m})$
2: for $(i = 0; i < \bar{m}; i \leftarrow i + 1)$ do
3:     for $(j = 0; j < \bar{n}; j \leftarrow j + 1)$ do
4:         $\mathbf{W}_{i,j} = GrayE(\mathbf{u}^{(i,j)})$
5:     end for
6: end for
7: Return $\mathbf{K} = \frac{q}{2^h} \cdot \mathbf{W} = \frac{q}{2^h} \cdot (\mathbf{W}_{i,j})_{0\le i<\bar{m}, 0\le j<\bar{n}}$
```

Algorithm 2 MsgDec

Input: Matrix $\mathbf{K} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$.

Output: Message $\boldsymbol{m} \in \{0,1\}^{l_e}$

```
1: for $(i = 0; i < \bar{m}; i \leftarrow i + 1)$ do
2:     for $(j = 0; j < \bar{n}; j \leftarrow j + 1)$ do
3:         $\mathbf{W}_{i,j} \leftarrow \lfloor \frac{2^h}{q} \cdot \mathbf{K}_{i,j} \rceil$
4:         $(\mathbf{u}_{(i+\bar{m}j)h}, \mathbf{u}_{(i+\bar{m}j)h+1}, \cdots, \mathbf{u}_{(i+\bar{m}j)h+h-1}) \leftarrow GrayD(\mathbf{W}_{i,j})$
5:     end for
6: end for
7: $\boldsymbol{m} \leftarrow BCD(\boldsymbol{u})$
8: Return $\boldsymbol{m}$
```

applying the inverse of the three steps. In fact, the error correction capacity can be characterized by the following lemma.

Lemma 2. *For any* $\boldsymbol{m} \in \{0,1\}^{l_e}$ *and* $\mathbf{E} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$*, it has* $\mathit{MsgDec}(\mathit{MsgEnc}(\mathbf{m}) + \mathbf{E}) = \mathbf{m}$ *provided*

$$\sum_{\substack{0\le i<\bar{m}\\ 0\le j<\bar{n}}} \left| \lfloor \frac{2^h}{q} \cdot \mathbf{E}_{i,j} \rceil \right| \le t_e. \tag{1}$$

Proof. Denote $\mathbf{u} = \mathit{BCE}(\mathbf{m}), \mathbf{W} = \mathit{GrayE}(\mathbf{u}), \mathbf{W}' = \lfloor \frac{2^h}{q} \cdot (\mathit{MsgEnc}(\mathbf{m}) + \mathbf{E}) \rceil$, and $\mathbf{u}' = \mathit{GrayD}(\mathbf{W}')$. Since

$$\begin{aligned} \mathbf{W}' &= \lfloor 2^h/q \cdot (\mathit{MsgEnc}(\mathbf{m}) + \mathbf{E}) \rceil \\ &= 2^h/q \cdot \mathit{MsgEnc}(\mathbf{m}) + \lfloor 2^h/q \cdot \mathbf{E} \rceil \\ &= \mathbf{W} + \lfloor 2^h/q \cdot \mathbf{E} \rceil, \end{aligned}$$

then $\sum_{\substack{0\le i<\bar{m}\\ 0\le j<\bar{n}}} |\mathbf{W}_{i,j} - \mathbf{W}'_{i,j}| \mod 2^h \le t_e$. Therefore $w_H(\mathbf{u} - \mathbf{u}') \le t_e$ according to the property of Gray code. It follows that $\mathit{BCD}(\mathbf{u}') = \mathbf{m}$. □

3.3 Construction of IND-CPA PKE

Our IND-CPA secure PKE scheme, which is denoted by SCloudCPAPKE, is described in Figure 1. The scheme includes three algorithms, i.e., SCouldPKE.KeyGen, SCloudCPAPKE.Enc and SCloudCPAPKE.Dec, which are shown in Algorithm 3, Algorithm 4 and Algorithm 5 respectively.

$$
\begin{array}{lcl}
\text{Initiator} & & \text{Responder} \\
\\
\text{KeyGen()} \rightarrow (pk, sk): & & \\
\mathbf{seed_A} \leftarrow \{0,1\}^{\kappa} & & \\
\mathbf{A} = Gen(\mathbf{seed_A}) \in \mathbb{Z}_q^{n\times n} & & \\
\mathbf{S}, \mathbf{E} \leftarrow \chi^{n\times \bar{n}} & & \\
\mathbf{B} = \mathbf{AS} + \mathbf{E} & & \text{Enc}(\mathbf{m} \in \mathbb{Z}^{l_e}, pk) \rightarrow c: \\
sk \leftarrow \mathbf{S};\ pk \leftarrow (\mathbf{seed_A}, \mathbf{B}) & \xrightarrow{pk} & \mathbf{A} = Gen(\mathbf{seed_A}) \\
 & & \mathbf{S}', \mathbf{E}' \leftarrow \chi^{\bar{m}\times n}; \mathbf{E}'' \leftarrow \chi^{\bar{m}\times \bar{n}} \\
 & & \mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}' \\
\text{Dec}(c, sk) \rightarrow \mathbf{m}: & \xleftarrow{c} & \mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}'' + MsgEnc(\mathbf{m}) \in \mathbb{Z}^{\bar{m}\times \bar{n}} \\
\mathbf{D} = \mathbf{V} - \mathbf{B}'\mathbf{S} & & c \leftarrow (\mathbf{B}', \mathbf{V}) \\
\mathbf{m} = MsgDec(\mathbf{D}) & &
\end{array}
$$

Figure 1: IND-CPA PKE scheme SCloudCPAPKE

Algorithm 3 SCloudCPAPKE.KeyGen

Input: None.

Output: (pk, sk).

```
1: Choose a uniformly random seed $\mathbf{seed_A}$
2: Generate matrix uniformly at random $\mathbf{A} \leftarrow Gen(\mathbf{seed_A}) \in \mathbb{Z}_q^{n\times n}$
3: Sample the private key and error matrixs $\mathbf{S}, \mathbf{E} \leftarrow \chi^{n\times \bar{n}}$
4: Compute $\mathbf{B} = \mathbf{AS} + \mathbf{E}$
5: Return the public key $pk \leftarrow (\mathbf{seed_A}, \mathbf{B})$, and secret key $sk \leftarrow \mathbf{S}$
```

Algorithm 4 SCloudCPAPKE.Enc

Input: Message $\mathbf{m} \in \mathbb{Z}^{l_e}$, public key $pk = (\mathbf{seed_A}, \mathbf{B})$.

Output: Ciphertext c.

```
1: Generate matrix uniformly at random $\mathbf{A} \leftarrow Gen(\mathbf{seed_A}) \in \mathbb{Z}_q^{n\times n}$
2: Generate error matrix $\mathbf{S}', \mathbf{E}' \leftarrow \chi^{\bar{m}\times n}$, $\mathbf{E}'' \leftarrow \chi^{\bar{m}\times \bar{n}}$
3: Compute $\mathbf{B}' = \mathbf{S}'\mathbf{A} + \mathbf{E}'$ and $\mathbf{V} = \mathbf{S}'\mathbf{B} + \mathbf{E}'' + MsgEnc(\mathbf{m})$
4: Return the ciphertext $c \leftarrow (\mathbf{B}', \mathbf{V})$
```

Algorithm 5 SCloudCPAPKE.Dec

Input: Ciphertext $c = (\mathbf{B}', \mathbf{V}) \in \mathbb{Z}_q^{\bar{m}\times n} \times \mathbb{Z}_q^{\bar{m}\times \bar{n}}$, secret key $sk = \mathbf{S} \in \mathbb{Z}_q^{n\times \bar{n}}$.

Output: Message $\mathbf{m}$.

```
1: Compute $\mathbf{D} = \mathbf{V} - \mathbf{B}'\mathbf{S}$
2: Return message $\mathbf{m} = MsgDec(\mathbf{D})$
```

3.3.1 Correctness

The decryption algorithm SCloudCPAPKE.Dec computes:

$$\begin{aligned}\mathbf{V} - \mathbf{B}'\mathbf{S} &= \mathbf{S}'\mathbf{B} + \mathbf{E}'' + MsgEnc(\mathbf{m}) - (\mathbf{S}'\mathbf{A}\mathbf{S} + \mathbf{E}'\mathbf{S}) \\ &= \mathbf{S}'\mathbf{A}\mathbf{S} + \mathbf{S}'\mathbf{E} + \mathbf{E}'' + MsgEnc(\mathbf{m}) - \mathbf{S}'\mathbf{A}\mathbf{S} - \mathbf{E}'\mathbf{S} \\ &= \mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S} + MsgEnc(\mathbf{m}).\end{aligned}$$

Write $\mathbf{E}''' = \mathbf{S}'\mathbf{E} + \mathbf{E}'' - \mathbf{E}'\mathbf{S}$, then by Lemma 2 the scheme decrypt successfully if and only if $\mathbf{E}'''$ satisfies the condition in (1).

Note that every single element of the matrix $\mathbf{E}'''$ can be viewed as the sum of $2n + 1$ independent random variables. Among them, $2n$ variables are the product of two independent Gaussian variables generated χ, and the rest one is a Gaussian variable generated by χ. We write the sum of these $2n+1$ variables as the variable χ', the decryption failure probability can be expressed as

$$p_{error} = 1 - \sum_{\sum |\lfloor (2^h/q)\cdot \mathbf{E}'''_{i,j} \rceil| \le t} \prod_{i,j} \chi'(\mathbf{E}'''_{i,j}).$$

3.4 Construction of IND-CCA KEM

In this subsection, we use the Fujisaki-Okamoto transformation [14] and obtain our IND-CCA scheme from a one-way-secure PKE scheme in the classical random oracle model. Particularly, we follow the approach adopted in [25], and construct an IND-CCA KEM scheme by using SCloudCPAPKE and two hash functions. Our KEM scheme SCloudKEM is described in Figure 2. The scheme includes algorithms SCloudKEM.KeyGen, SCloudKEM.Encaps and S-CloudKEM.Decaps, which are in shown in Algorithm 6, Algorithm 7 and Algorithm 8 respectively.

```
Initiator                                                  Responder

KeyGen()$\rightarrow (pk, sk')$:
$(pk, sk) \leftarrow$ PKE.KenGen()             $\xrightarrow{pk}$
$\mathbf{s} \leftarrow \{0,1\}^l$                                 Encaps($pk$)$\rightarrow (\mathbf{ss}, \mathbf{B}', \mathbf{C}, \mathbf{d})$:
$sk' \leftarrow (sk, pk, \mathbf{s})$                             $\mathbf{m} \leftarrow \{0,1\}^k$
                                                           $\mathbf{r}, \mathbf{k}, \mathbf{d} \leftarrow G(pk||\mathbf{m})$
                                                           $(\mathbf{B}', \mathbf{C}) \leftarrow$ PKE.Enc$(\mathbf{m}, pk; \mathbf{r})$

Decaps($sk', \mathbf{B}', \mathbf{C}, \mathbf{d}$)$\rightarrow \mathbf{ss}'$:     $\xleftarrow{(\mathbf{B}',\mathbf{C},\mathbf{d})}$     $\mathbf{ss} \leftarrow F(\mathbf{B}'||\mathbf{C}||\mathbf{k}||\mathbf{d})$
$\mathbf{m}' \leftarrow$ PKE.Dec$(\mathbf{B}', \mathbf{C}, sk)$
$(\mathbf{r}', \mathbf{k}', \mathbf{d}') \leftarrow G(pk||\mathbf{m}')$
if $(\mathbf{B}', \mathbf{C}) =$ PKE.Enc$(\mathbf{m}', pk; \mathbf{r}')$
    and $\mathbf{d} = \mathbf{d}'$
  then $\mathbf{ss}' \leftarrow F(\mathbf{B}'||\mathbf{C}||\mathbf{k}'||\mathbf{d})$
  else $\mathbf{ss}' \leftarrow F(\mathbf{B}'||\mathbf{C}||\mathbf{s}||\mathbf{d})$
```

Figure 2: IND-CCA KEM scheme SCloudKEM.

Algorithm 6 SCloudKEM.KeyGen

```
Input: None.
Output: Public and secret key $(pk, sk')$.
1: Generate uniformly random seed $\mathbf{s}||\mathbf{seed_E}||\mathbf{z}$
2: Generate pseudorandom seed $\mathbf{seed_A} \leftarrow H(\mathbf{z})$
3: Use random seed $\mathbf{seed_A}$ to generate matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times n}$: $\mathbf{A} \leftarrow Gen(\mathbf{seed_A})$
4: Use random seed $\mathbf{seed_E}$ to sample matrixs $\mathbf{S}, \mathbf{E} \leftarrow \chi^{n\times\bar{n}}$
5: Compute $\mathbf{B} \leftarrow \mathbf{AS} + \mathbf{E}$
6: Return $pk \leftarrow (\mathbf{seed_A}, \mathbf{B})$ and $sk' \leftarrow (\mathbf{s}, \mathbf{seed_A}, \mathbf{B}, \mathbf{S})$
```

Algorithm 7 SCloudKEM.Encaps()

```
Input: $pk = (\mathbf{seed_A}, \mathbf{B})$.
Output: Ciphertext $(\mathbf{B}', \mathbf{C}, \mathbf{d})$.
1: Choose uniformly random message $\mathbf{m}$
2: Generate pseudorandom values $\mathbf{r}, \mathbf{k}, \mathbf{d} \leftarrow G(\mathbf{seed_A}||\mathbf{B}||\mathbf{m})$
3: Use random seed $\mathbf{r}$ to sample matrixs $\mathbf{S}', \mathbf{E} \leftarrow \chi^{\bar{m}\times n}$,$\mathbf{E}'' \leftarrow \chi^{\bar{m}\times\bar{n}}$
4: Generate matrix $\mathbf{A} \leftarrow Gen(\mathbf{seed_A})$
5: Compute $\mathbf{B}' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}$
6: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
7: Compute $\mathbf{C} \leftarrow \mathbf{V} + MsgEnc(\mathbf{m})$
8: Compute $\mathbf{ss} \leftarrow F(\mathbf{B}'||\mathbf{C}||\mathbf{k}||\mathbf{d})$
9: Return the ciphertext $\mathbf{B}', \mathbf{C}, \mathbf{d}$ and shared session key $\mathbf{ss}$
```

Algorithm 8 SCloudKEM.Decaps()

```
Input: Ciphertext $\mathbf{B}', \mathbf{C}, \mathbf{d}$, secret key $sk' \leftarrow (\mathbf{s}, \mathbf{seed_A}, \mathbf{B}, \mathbf{S})$.
Output: Agreement session key $\mathbf{ss}$.
 1: Compute $\mathbf{D} = \mathbf{C} - \mathbf{B}'\mathbf{S}$
 2: Compute $\mathbf{m}' \leftarrow MsgDec(\mathbf{D})$
 3: Generate pseudorandom values $\mathbf{r}', \mathbf{k}', \mathbf{d}' \leftarrow G(\mathbf{seed_A}||\mathbf{B}||\mathbf{m}')$
 4: Use random seed $\mathbf{r}'$ to generate error matrix $\mathbf{S}', \mathbf{E}' \leftarrow \chi^{\bar{m}\times n}$,$\mathbf{E}'' \leftarrow \chi^{\bar{m}\times\bar{n}}$
 5: Generate matrix $\mathbf{A} \leftarrow Gen(\mathbf{seed_A})$
 6: Compute $\mathbf{B}'' \leftarrow \mathbf{S}'\mathbf{A} + \mathbf{E}'$
 7: Compute $\mathbf{V} \leftarrow \mathbf{S}'\mathbf{B} + \mathbf{E}''$
 8: Compute $\mathbf{C}' \leftarrow \mathbf{V} + MsgEnc(\mathbf{m}')$
 9: if $(\mathbf{B}'||\mathbf{C} = \mathbf{B}''||\mathbf{C}')$ and $(\mathbf{d} = \mathbf{d}')$ then
10:     return $\mathbf{ss} \leftarrow F(\mathbf{B}'||\mathbf{C}||\mathbf{k}'||\mathbf{d})$
11: else
12:     return $\mathbf{ss} \leftarrow F(\mathbf{B}'||\mathbf{C}||\mathbf{s}||\mathbf{d})$
13: end if
```

3.4.1 Correctness

The decryption failure probability of IND-CCA SCloudKEM is the same as that of the previous IND-CPA SCloudCPAPKE.

3.5 Construction of IND-CCA PKE

Our IND-CCA secure PKE scheme SCloudCCAPKE is constructed based on the IND-CCA secure KEM scheme SCloudKEM and a data-encapsulation mechanism (DEM), which follows from the approach proposed by Cramer and Shoup[13]. The detail of the scheme can be found in Figure 3 and the algorithms SCloudCCAPKE.KeyGen, SCloudCCAPKE.Enc and SCloudCCAPKE.Dec are specified in Algorithm 9, Algorithm 10 and Algorithm 11 respectively.

Algorithm 9 SCloudCCAPKE.KeyGen

```
Input: None.
Output: Public and secret keys $(pk, sk')$.
 1: Use SCloudKEM.KeyGen() to generate public and secret keys $(pk, sk')$
 2: Return public key $pk$ and secret key $sk'$
```

Initiator		Responder
KeyGen()$\rightarrow$ (pk, sk'):		
(pk, sk') $\leftarrow$KEM.KenGen()	$\xrightarrow{pk}$	
		Enc($\mathbf{ms} \in \mathbb{Z}^{l_m}$,$pk$)$\rightarrow$ $(\mathbf{B}', \mathbf{C}, \mathbf{d}, \mathbf{K})$
		$(\mathbf{ss}, \mathbf{B}', \mathbf{C}, \mathbf{d})$ $\leftarrow$KEM.Encaps(pk)
Decaps($sk', \mathbf{B}', \mathbf{C}, \mathbf{d}, \mathbf{K}$)$\rightarrow$ $\mathbf{ms}'$		$c2 = \text{DEM}(\mathbf{ss}, \mathbf{ms})$
$\mathbf{ss}'$ $\leftarrow$KEM.Decaps($\mathbf{B}', \mathbf{C}, \mathbf{d}, sk'$)	$\xleftarrow{(\mathbf{B}',\mathbf{C},\mathbf{d},c2)}$	
$\mathbf{ms} \leftarrow \text{DEM}^{-1}(\mathbf{ss}', c2)$		

Figure 3: IND-CCA PKE scheme SCloudCCAPKE

Algorithm 10 SCloudCCAPKE.Enc

```
Input: Message $\mathbf{ms} \in \mathbb{Z}^{l_m}$, public key $pk$.
Output: Ciphertext $c$.
1: Use SCloudKEM.Encaps($pk$) to encapsulate $(\mathbf{ss}, \mathbf{B}', \mathbf{C}, \mathbf{d})$
2: Use DEM to encapsulate encryption message $c2 = \text{DEM}(\mathbf{ss}, \mathbf{ms})$
3: Return ciphertext $c \leftarrow (\mathbf{B}', \mathbf{C}, \mathbf{d}, c2)$
```

Algorithm 11 SCloudCCAPKE.Dec

```
Input: Ciphertext $c = (\mathbf{B}', \mathbf{C}, \mathbf{d}, c2)$, secret key $sk'$.
Output: Decryption message $\mathbf{ms}$.
1: Use SCloudKEM.Decaps($\mathbf{B}', \mathbf{C}, \mathbf{d}, sk'$) to decapsulate $\mathbf{ss}'$
2: Use $\text{DEM}^{-1}(\mathbf{ss}', c2)$ to decrypt $\mathbf{ms}$
3: Return message $\mathbf{ms}$
```

3.6 Cryptographic Primitives

The following cryptographic primitives are involved in our schemes, of which the security requirement is specified below.

- Gen: A public function used to generate the pseudorandom matrix $\mathbf{A}$. In our scheme, we recommend SM3 [30] hash algorithm as Gen function.
- G: A pseudorandom number generator to generate the random seeds required in KEM. We recommend SM3 [30] hash algorithm as G function.
- F: A pseudorandom number generator to generate the final agreement session key in KEM. We recommend SM3 [30] hash algorithm as F function.
- H: A pseudorandom number generator to generate random seed in KEM and PKE. We recommend SM3 [30] hash algorithm as H function.
- DEM: We use (AES-GCM)— encryption algorithm as DEM.

4 Parameters

In this section, we provide the following four sets of parameters for different security level and message length of SCloud.

- High efficiency & Long message mode:
  SCloud-f640, SCloud-f896 and SCloud-f1216 (see Table 1).
- High efficiency & Short message mode:
  SCloud-fg640, SCloud-fg896 and SCloud-fg1216 (see Table 2).
- High security & Long message mode:
  SCloud-q640, SCloud-q896 and SCloud-q1216 (see Table 3).
- High security & Short message mode:
  SCloud-qg640, SCloud-qg896 and SCloud-qg1216 (see Table 4).

High security mode provides parameters that satisfy the LWE reduction condition proposed in [27, 26] ($\sigma \geq \frac{\sqrt{n}}{2\pi}$) so as to keep security against known attacks including primal attack, dual attack, BKW and algebra attack.

High efficiency mode provides better performance and smaller size with practical security against primal attack and dual attack which are the most common methods considered in lattice-based designs.

Long message mode refers to the message length is the double security level (in bits), which takes into consideration the quantum acceleration of the exhaustive search of message.

Short message mode refers to the message length is equal to the security level (in bits).

Table 1: Parameters of high efficiency & long message mode

	f640	f896	f1216
security level(bit)	128	192	256
n	640	896	1216
q	2^{14}	2^{14}	2^{15}
$(\bar{m}, \bar{n})$	$(9, 11)$	$(12, 14)$	$(13, 15)$
σ	2.12	2.12	2.12
length of message m(bit)	256	384	512
(k_1, k_2) in χ_{k_1,k_2}	(5,3)	(5,3)	(5,3)
decryption failure probability	2^{-135}	2^{-206}	2^{-295}
$[n_e, l_e, d_e]$ binary linear code	[292,256,9]	[474,384,21]	[552,512,9]
h-bit Gray Code	3	3	3

Table 2: Parameters of high efficiency & short message mode

	fg640	fg896	fg1216
security level(bit)	128	192	256
n	640	896	1216
q	2^{14}	2^{14}	2^{15}
$(\bar{m}, \bar{n})$	$(7, 8)$	$(8, 12)$	$(9, 11)$
σ	2.12	2.12	2.12
message$m$	128	192	256
(k_1, k_2) in χ_{k_1,k_2}	(5,3)	(5,3)	(5,3)
decryption failure probability	2^{-168}	2^{-214}	2^{-299}
$[n_e, l_e, d_e]$ binary linear code	[168,128,11]	[282,192,21]	[292,256,9]
h-bit Gray code	3	3	3

Table 3: Parameters of high security & long message mode

	q640	q896	q1216
security level(bit)	128	192	256
n	640	896	1216
q	2^{16}	2^{17}	2^{18}
$(\bar{m}, \bar{n})$	$(9, 11)$	$(12, 12)$	$(13, 14)$
σ	4.10	4.83	5.85
message m	256	384	512
(k_1, k_2) in χ_{k_1,k_2}	(15,14)	(20,20)	(30,29)
decryption failure probability	2^{-135}	2^{-206}	2^{-295}
$[n_e, l_e, d_e]$ binary linear code	[292,256,9]	[474,384,21]	[552,512,9]
h-bit Gray code	3	3	3

5 Security Analysis

5.1 Security Reductions

5.1.1 IND-CPA Security of SCloudCPAPKE

Theorem 1. *Let $n, q, \bar{m}, \bar{n}$ be positive integers, and let χ be a distribution on $\mathbb{Z}$. Assume that the matrix $\mathbf{A} \leftarrow U(\mathbb{Z}_q^{n\times n})$. For any quantum algorithm $\mathcal{A}$ against the IND-CPA security of* SCloudCPAPKE, *there exist quantum algorithms $\mathcal{F}_1$ and*

Table 4: Parameters of high security & short message mode

	qg640	qg896	qg1216
security level(bit)	128	192	256
n	640	896	1216
q	2^{16}	2^{17}	2^{18}
$(\bar{m}, \bar{n})$	$(6, 9)$	$(8, 9)$	$(8, 12)$
σ	4.10	4.83	5.85
message$m$	128	192	256
(k_1, k_2) in χ_{k_1,k_2}	(15,14)	(20,20)	(30,29)
decryption failure probability	2^{-128}	2^{-192}	2^{-264}
$[n_e, l_e, d_e]$ binary linear code	[152,128,7]	[216,192,7]	[283,256,7]
h-bit Gray code	3	3	3

$\mathcal{F}_2$ against the decision version of the normal-form LWE *such that*

$$\mathbf{Adv}^{ind-cpa}_{SCloudCPAPKE}(\mathcal{A}) \leq \bar{n} \cdot \mathbf{Adv}^{nf-dlew}_{n,q,\chi}(\mathcal{F}_1) + \bar{m} \cdot \mathbf{Adv}^{nf-dlew}_{n,n+\bar{n},q\chi}(\mathcal{F}_2).$$

where the running time of $\mathcal{F}_1$ and $\mathcal{F}_2$ are approximately that of $\mathcal{A}$。

Proof. The proof is similar to that of Theorem 1 in [22]. □

5.1.2 IND-CCA Security of SCloudKEM

Theorem 2. *Let* PKE *be a δ-correct PKE scheme, which includes algorithms* (KeyGen, Enc, Dec) *with message space $\mathcal{M}$. Let G and F be independent random oracles. Assume $\mathcal{A}$ to be any classical algorithm against the IND-CCA security of* SCloudKEM, *and $\mathcal{A}$ makes q_G and q_F queries of its G and F. Then there exists a classical algorithm $\mathcal{F}$ against the IND-CPA security of* PKE *such that*

$$\mathbf{Adv}^{ind-cca}_{SCloudKEM}(\mathcal{A}) \leq \frac{4 \cdot q_{RO} + 1}{|\mathcal{M}|} + q_{RO} \cdot \delta + 3 \cdot \mathbf{Adv}^{ind-cpa}_{PKE}(\mathcal{F})$$

where $q_{RO} = q_G + q_F$. The running time of $\mathcal{F}$ is about that of $\mathcal{A}$。

Theorem 3. *Let* PKE *be a δ-correct PKE scheme, which includes* (KeyGen, Enc, Dec) *with message space $\mathcal{M}$. Let G and F be independent random oracles. Assume*

$\mathcal{A}$ to be any quantum algorithm against the IND-CCA security of SCloudKEM*, and $\mathcal{A}$ makes q_G and q_F queries of its G and F. Then there exists a quantum algorithm $\mathcal{F}$ against the IND-CPA security of* PKE *such that*

$$\mathbf{Adv}_{SCloudKEM}^{ind-cca}(\mathcal{A}) \leq 9 \cdot q_{RO} \cdot \sqrt{q_{RO}^2 \cdot \delta + q_{RO} \cdot \sqrt{\mathbf{Adv}_{PKE}^{ind-cpa}(\mathcal{F}) + \frac{1}{|\mathcal{M}|}}}$$

where $q_{RO} = q_G + q_F$. Here the running time of $\mathcal{F}$ is about that of $\mathcal{A}$.

Proof of Theorem 2 and Theorem 3. The proofs are similar to that of Theorem 5.1 and Theorem 5.2 in [25]. □

5.1.3 IND-CCA Security of SCloudCCAPKE

Note that our SCloudCCAPKE is constructed based on SCloudCCAKEM via the approach proposed by Cramer and Shoup [13], then its IND-CCA security is guaranteed by the IND-CCA security of SCloudCCAKEM. A similar deduction can also be found in [5].

5.2 Cryptanalytic Attacks

In this section, we introduce the commonly used methods of analyzing the LWE problem and then estimate the security of our scheme. The methods can be divided into two types according to the number of LWE samples that an attacker can obtain. When only $O(n)$ LWE samples are available, the BKZ-type attacks are usually taken into consideration, namely primal attack and dual attack. If more samples are available, the combinational-type attacks, namely algebraic attack [4] and BKW attack [19], are also calculated where $O(n^c)(c > 1)$ and $2^{O(n)}$ samples are needed respectively. Next, we introduce the two BKZ-type attacks. The algebraic attack and BKW attack are given in Appendix C, D.

5.2.1 BKZ-type Attacks

The BKZ [10] algorithm can be seen as a generalized version of LLL algorithm, which iteratively searches the shortest vector in the b-dimensional space instead of the 2-dimensional space. Therefore, it outputs vectors of better quality (that is, shorter vectors are outputted) than LLL algorithm. In terms of the running time, LLL algorithm can be strictly proved to be a polynomial time algorithm, however, the running time of BKZ algorithm is closely related to the block size b and it may need exponential operations.

It is difficult to estimate the exact running time of BKZ algorithm. The BKZ algorithm with block size b needs to call the b-dimensional SVP oracle polynomial times iteratively. According to the result of [17], the basis outputted by a BKZ algorithm after $n^2 \log n/b^2$ round iterations is closed to the final result of the algorithm. Taking the possible improvements of algorithms into consideration, a popular way to estimate the running time of BKZ algorithm in the attack is to consider the time of running b-dimensional SVP oracle, i.e. the "core-SVP" complexity is considered which is very conservative.

Among all konwn algorithms that can solve the b-dimensional SVP, the algorithm with lowest asymptotic complexity is sieving algorithm with its asymptotic complexity being $2^{cb+o(b)}$(where c is a constant and it has been widely studied). For the classical sieving algorithm, the constant is $c_C = \log_2 \sqrt{3/2} \approx 0.292$ which is obtained in [7]; for the quantum algorithms, this constant is $c_Q = \log_2 \sqrt{13/9} \approx 0.265$. Since all the variants of sieving algorithm need to establish a vector list with size $(\sqrt{4/3})^b$, we can regard $c_P = \log_2 \sqrt{4/3} \approx 0.2075$ as the lowest bound of the time complexity of plausible sieving algorithm. When estimating the time of sieving algorithm, the item $o(b)$ is ignored.

Notice that the above estimation is very conservative compared with the actual application of [24]. In the practical experiment, the complexity of sieve algorithm is about $2^{0.405b+11}$ when the block size b is between $[60, 80]$.

5.2.2 Primal Attack

The primal attack is to construct a lattice with the unique shortest vector from an instance of LWE problem, and then use BKZ to solve the unique shortest vector problem. Concretely, given a LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{As}+\mathbf{e})$, we can construct a d-dimensional (for $d = m+n+1$) lattice $\Lambda = \{\mathbf{x} \in \mathbb{Z}^{m+n+1} : (\mathbf{A}|\mathbf{I}_m|-\mathbf{b})\mathbf{x} = \mathbf{0}\ mod\ q\}$ with volume q^m. It is easy to see that the unique shortest vector in Λ is $\mathbf{v} = (\mathbf{s}, \mathbf{e}, 1)$ with length $\lambda \approx \sigma\sqrt{n+m}$. $m \in [0, 2n]$ is the number of samples used and its specific value can be determined by the optimization of the attack. Therefore, combined with the above-mentioned complexity of the core-SVP, we only need to estimate the smallest block size used in BKZ algorithm.

Based on the well known GSA assumption of BKZ algorithm, we can treat $\{\|\mathbf{b}_i^*\|\}_{i=0}^{d-1}$ as the geometric series $\|\mathbf{b}_i^*\| = \delta^{d-2i-1} \cdot vol(\mathcal{L})^{1/d}$. If the projection length of the shortest vector $\mathbf{v}$ in the last b-dimensional space is shorter than $\|\mathbf{b}_{d-b}^*\|$, then it can be obtained by BKZ algorithm. Therefore, it can be concluded that the primal attack is successful if $\sigma\sqrt{b} \leq \delta^{2b-d-1} \cdot q^{m/d}$, where the parameter $\delta = ((\pi b)^{1/b}\frac{b}{2\pi e})^{1/(2(b-1))}$. Moreover, we can get the smallest block size b, and then the complexity estimation of the BKZ algorithm is obtained by combining the block size with the difficulty of the core-SVP.

5.2.3 Dual Attack

The dual attack is a method to solving decision-LWE and its core idea can be concluded as following: first find the short vectors in dual lattice $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2) \in \Lambda' = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^m \times \mathbb{Z}^n : \mathbf{A}^t\mathbf{x} = \mathbf{y}\ mod\ q\}$ by BKZ algorithm, then use the short vectors $\mathbf{w}$ to distinguish the LWE sample, i.e. calculate the value $z := \mathbf{w}_1^t \cdot \mathbf{b} = \mathbf{w}_1^t\mathbf{As} + \mathbf{w}_1^t\mathbf{e} = \mathbf{w}_2^t\mathbf{s} + \mathbf{w}_1^t\mathbf{e}\ mod\ q$.

Notice that the dimension of dual lattice Λ' is $d = m + n$ and its volume is q^n. A BKZ algorithm with block size b outputs the short vector with length $l = \|\mathbf{b}_0\| = \delta^{d-1}q^{n/d}$. If $(\mathbf{A}, \mathbf{b})$ is an instance of LWE distribution, then $z = \mathbf{w}_2^t\mathbf{s} + \mathbf{w}_1^t\mathbf{e}\ mod\ q$ follows a Gaussian distribution with standard deviation $l\sigma$, else z follows $U(\mathbb{Z}_q)$.

The maximum distance between two distributions is $\varepsilon = 4\exp(-2\pi^2\tau^2),\ \tau = l\sigma/q$. Therefore an attacker can distinguish LWE samples from random samples with advantage ε.

The small advantages ε are not relevant and an attacker needs an advantage of at least $1/2$ to significantly decrease the search space. Therefore, we needs about $1/\varepsilon^2$ short vectors. Since running a sieving algorithm once can provide $2^{0.20756b}$ vectors, the attack needs to be repeated at least R times where $R = max(1, 1/(2^{20756b}\varepsilon^2))$. In short, the total complexity of dual attack is $R \cdot 2^{cb}$ and attacker needs to choose optimal b to make the complexity as small as possible.

In Table 5, the complexity of primal attack and dual attack with efficiency parameters are given. In this table, the 'primal' represents the primal attack, the 'dual' represents the dual attack, the 'classical' represents the known classical complexity, the 'quantum' represents the known quantum complexity, the 'plausible' represents the best possible algorithm complexity, the 'm' represents the number of samples required, and the 'b' denotes the block size of BKZ algorithm.

Table 5: The hardness of efficiency parameters

parameters (n, q, σ)	attack	(m,b)	classical	quantum	plausible
f640:$(640, 2^{14}, 2.12)$	primal	(655,495)	145	131	103
	dual	(737,489)	143	130	101
f896:$(896, 2^{14}, 2.12)$	primal	(924,741)	216	196	153
	dual	(972,734)	214	195	153
f1216:$(1216, 2^{15}, 2.12)$	primal	(1199,976)	285	259	203
	dual	(1297,968)	283	257	201

6 Performance Analysis

The PKE scheme SCloudCPAPKE includes three algorithms named key generation, encryption and decryption. Their performances can be calculated by considering the complexities of calling random generator, making matrix operations and running encode/decode process.

In the key generation process, the random generator needs to output n^2 random integers in $\mathbb{Z}_q$ and support $2n\bar{n}$ samplings. For matrix operations, we need a matrix multiplication between one matrix of the size $n \times n$ and the other of the size $n \times \bar{n}$ as well as a matrix addition between two $n \times \bar{n}$ matrixes. As a result, in the key generation process, we need $n^2\bar{n}$ multiplications, $n^2\bar{n}$ additions and $n^2 \log q + 2n\bar{n} \times E_\chi$ random bits where E_χ denotes the expectation number of bits needed to sample from the distribution χ.

In the encryption process, the random generator needs to output n^2 random integers in $\mathbb{Z}_q$ and support $2\bar{m}n + \bar{m}\bar{n}$ samplings. For matrix operations, we need a matrix multiplication between one matrix of the size $\bar{m} \times n$ and the other of the size $n \times n$, a matrix multiplication between one matrix of the size $\bar{m} \times n$ and the other of the size $n \times \bar{n}$, a matrix addition between two $\bar{m} \times n$ matrixes and a matrix addition between two $\bar{m} \times \bar{n}$ matrixes. Besides, an encoding process should be conducted. As a result, in the encryption process, we need $n^2\bar{m} + \bar{m}\bar{n}n$ multiplications, $n^2\bar{m} + \bar{m}\bar{n}n$ additions, $n^2 \log q + (2\bar{m}n + \bar{m}\bar{n}) \times E_\chi$ random bits combined with an encoding call which needs about $l_e(n_e - l_e)$ Xor operations, $\bar{m}\bar{n}$ table check and $\bar{m}\bar{n}$ multiplications.

In the decryption process, we need a matrix multiplication between one matrix of the size $\bar{m} \times n$ and the other of the size $n \times \bar{n}$, a matrix addition between two $\bar{m}\bar{n}n$ matrixes and a decoding process. As a result, in the decryption process, we need $\bar{m}\bar{n}n$ multiplications, $\bar{m}\bar{n}n$ additions combined with a decoding call which needs about $\bar{m}\bar{n}$ table check, $\bar{m}\bar{n}$ multiplications and $2 \cdot m_{ext}n_e(d_e - 1)$ Xor operations [6] where $m_{ext} = 8$ for the parameters fg640, qg640, qg896 and $m_{ext} = 9$ for other parameter sets.

It should be remarked that the efficiency of generating random bits are decided by the performance of random source. When taking SM3 as the random source, we need 2-5 clock cycles to obtain a random bit, and we will use 5 clock cycles/bit in the following computations [30].

The performance of SCloudCPAPKE with high efficiency & long message parameters is shown in table 6 and its size can be found in Table 9.

The algorithms of SCloudKEM are derived by the three basic algorithms of SCloudCPAPKE whose performances can be computed similarly. We conclude performance of SCloudKEM using high efficiency parameters in table 7 and its size in Table 10. Besides, the scheme SCloudCCAPKE is designed by combing SCloudKEM with DEM where 256-bit AES-GCM is used, the performance of SCloudCCAPKE can be found in Table 8 and the size in Table 11.

Table 6: Performance of SCloudCPAPKE with high efficiency & long message mode

parameters	$f640$	$f896$	$f1216$
key generation(clock cycle)	3.88×10^7	8.07×10^7	1.58×10^8
encryption(clock cycle)	3.71×10^7	7.75×10^7	1.52×10^8
decryption(clock cycle)	2.46×10^4	3.99×10^4	5.65×10^5

Table 7: Performance of SCloudKEM with high efficiency & long message mode

parameters	$f640$	$f896$	$f1216$
key generation(clock cycle)	3.88×10^7	8.07×10^7	1.58×10^7
encapsulation(clock cycle)	3.71×10^7	7.75×10^7	1.52×10^8
decapsulation(clock cycle)	3.71×10^7	7.75×10^7	1.52×10^8

Table 8: Performance of SCloudCCAPKE with high efficiency & long message mode

parameters	$f640$	$f896$	$f1216$
key generation(clock cycle)	1.88×10^7	4.14×10^7	8.07×10^7
encryption(clock cycle)	1.71×10^7	3.82×10^7	7.48×10^7
decryption(clock cycle)	1.71×10^7	3.82×10^7	7.48×10^7

Table 9: Size of inputs and outputs of SCloudCPAPKE with high efficiency & long message mode

parameters	$f640$	$f896$	$f1216$
public key(bit)	9.88×10^4	1.76×10^5	2.76×10^5
secret key(bit)	9.86×10^4	1.76×10^5	2.74×10^5
ciphertext(bit)	8.2×10^4	1.53×10^5	2.40×10^5
message(bit)	256	384	512

Table 10: Size of inputs and outputs of SCloudKEM with high efficiency & long message mode

parameters	$f640$	$f896$	$f1216$
public key(bit)	9.88×10^4	1.76×10^5	2.74×10^5
secret key(bit)	1.98×10^5	3.52×10^5	5.48×10^5
ciphertext(bit)	8.23×10^4	1.53×10^5	2.40×10^5
shared secret(bit)	256	384	512

Table 11: Size of inputs and outputs of SCloudCCAPKE with high efficiency & long message mode

parameters	$f640$	$f896$	$f1216$
public key(bit)	9.88×10^4	1.76×10^5	2.74×10^5
secret key(bit)	1.98×10^5	3.52×10^5	5.48×10^5
ciphertext(bit)	8.25×10^4	1.53×10^5	2.41×10^5
message(bit)	256	384	512

7 The Advantages and Limitations

7.1 Advantages

- Quantum resistance: Our scheme is based on the LWE problem, which is proved at least as hard as the SVP problem on unstructured lattices. So far, the SVP problem over unstructured lattices is believed to be more difficult than that over structured lattices (e.g., the ideal lattices), and there is *no* efficient quantum algorithm to solve it up to now. So our scheme can be considered to be quantum resistant.
- Improved sampling algorithm: Compared with other sampling methods, our approach can achieve higher efficiency and more flexibility w.r.t the parameter choice. Besides, it is shown to be more secure against the dual attack due to its advantage in distinguish property.
- New error-reconciliation method: Long binary codes and Gray codes are combined in our new error-reconciliation scheme to improve both the transmission and computation efficiency. Moreover, our new method is more

adaptive w.r.t the choice of parameters.

7.2 Limitations

- Despite the advantages of using a powerful error-reconciliation method, the implementation of the involved BCH codes has to be constant time to avoid the timing attack, which may slightly affects the efficiency.

References

[1] Albrecht, M.R., Cid, C., Faugère, J., Fitzpatrick, R., Perret, L.: Algebraic algorithms for LWE problems. ACM Comm. Computer Algebra **49**(2), 62 (2015)

[2] Albrecht, M.R., Faugère, J., Fitzpatrick, R., Perret, L.: Lazy modulus switching for the BKW algorithm on LWE. In: Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings. pp. 429–445 (2014)

[3] Albrecht, M.R., Player, R., Scott, S.: On the concrete hardness of learning with errors. J. Mathematical Cryptology **9**(3), 169–203 (2015)

[4] Arora, S., Ge, R.: New algorithms for learning in presence of errors. In: Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I. pp. 403–415 (2011)

[5] Baan, H., Bhattacharya, S., Fluhrer, S., Garcia-Morchon, O., Laarhoven, T., Player, R., Rietman, R., Saarinen, M.J.O., Tolhuizen, L., Arce, J.L.T., Zhang, Z.: Round5: compact and fast post-quantum public-key encryption,https://round5.org/

[6] Bajoga, B.G., Walbesser, W.J.: Decoder complexity for bch codes. Proceedings of the Institution of Electrical Engineers **120**(4), 429–431 (2010)

[7] Becker, A., Ducas, L., Gama, N., Laarhoven, T.: New directions in nearest neighbor searching with applications to lattice sieving. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016. pp. 10–24 (2016)

[8] Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. Journal of the Acm **50**(4), 506–519

[9] Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. In: Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012. pp. 309–325 (2012)

[10] Chen, Y., Nguyen, P.Q.: BKZ 2.0: Better lattice security estimates. In: Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings. pp. 1–20 (2011)

[11] Chunhuan, Z., Zhongxiang, Z., Xiaoyun, W., Guangwu, X.: Distinguishing lwe instances using fourier transform: A refined framework and its applications.https://eprint.iacr.org/

[12] Cramer, R., Ducas, L., Wesolowski, B.: Short stickelberger class relations and application to ideal-svp. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. pp. 324–348 (2017)

[13] Cramer, R., Shoup, V.: Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. SIAM J. Comput. **33**(1), 167–226 (2003)

[14] Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. pp. 537–554 (1999)

[15] Guo, Q., Johansson, T., Mårtensson, E., Stankovski, P.: Coded-bkw with sieving. In: Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I. pp. 323–346 (2017)

[16] Guo, Q., Johansson, T., Stankovski, P.: Coded-bkw: Solving LWE using lattice codes. In: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. pp. 23–42 (2015)

[17] Hanrot, G., Pujol, X., Stehlé, D.: Analyzing blockwise lattice algorithms using dynamical systems. In: Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings. pp. 447–464 (2011)

[18] Hofheinz, D., Hövelmanns, K., Kiltz, E.: A modular analysis of the fujisaki-okamoto transformation. In: Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I. pp. 341–371 (2017)

[19] Kirchner, P., Fouque, P.: An improved BKW algorithm for LWE with applications to cryptography and lattices. In: Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. pp. 43–62 (2015). https://doi.org/10.1007/978-3-662-47989-6_3

[20] Kirchner, P., Fouque, P.: Revisiting lattice attacks on overstretched N-TRU parameters. In: Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I. pp. 3–26 (2017). https://doi.org/10.1007/978-3-319-56620-7_1, `https://doi.org/10.1007/978-3-319-56620-7_1`

[21] Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Des. Codes Cryptography **75**(3), 565–599 (2015)

[22] Lindner, R., Peikert, C.: Better key sizes (and attacks) for lwe-based encryption. In: Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011.

Proceedings. pp. 319–339 (2011). https://doi.org/10.1007/978-3-642-19074-2_21, https://doi.org/10.1007/978-3-642-19074-2_21

[23] Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Monaco / French Riviera, May 30 - June 3, 2010. Proceedings. pp. 1–23 (2010)

[24] Mariano, A., Laarhoven, T., Bischof, C.H.: A parallel variant of ldsieve for the SVP on lattices. In: 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing, PDP 2017, St. Petersburg, Russia, March 6-8, 2017. pp. 23–30 (2017)

[25] Naehrig, M., Alkim, E., Bos, J., Ducas, L., Easterbrook, K., LaMacchia, B., Longa, P., Mironov, I., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D.: Frodokem learning with errors key encapsulation. http://frodokem.org

[26] Peikert, C., Regev, O., Stephens-Davidowitz, N.: Pseudorandomness of ring-lwe for any ring and modulus. In: Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017. pp. 461–473 (2017)

[27] Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005. pp. 84–93 (2005)

[28] Schnorr, C.: A hierarchy of polynomial time lattice basis reduction algorithms. Theor. Comput. Sci. **53**, 201–224 (1987). https://doi.org/10.1016/0304-3975(87)90064-8, https://doi.org/10.1016/0304-3975(87)90064-8

[29] Stehlé, D., Steinfeld, R.: Making NTRU as secure as worst-case problems over ideal lattices. In: Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011.

Proceedings. pp. 27–47 (2011). https://doi.org/10.1007/978-3-642-20465-4_4, `https://doi.org/10.1007/978-3-642-20465-4_4`

[30] Wang, X. Yu, H.: Sm3 cryptographic hash algorithm. 2016

A The Distinguishing Advantages of Error Distribution

For a variable $\mathbf{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_n)$ where $\mathbf{x}_i \sim f(x)$ for $1 \leq i \leq n$ are independent and identically distributed variables over $\mathbb{Z}_q$ and a vector $\mathbf{v} \neq \mathbf{0} \in \mathbb{Z}_q^n$, we consider the new variable $\langle \mathbf{v}, \mathbf{X} \rangle$ and calculate its Fourier transform. According to the property of Fourier transform, we have

$$\hat{f}_{\langle \mathbf{v}, \mathbf{X} \rangle}(1) = \prod_{j=1}^{n} \hat{f}_{v_j \mathbf{x}_j}(1) = \prod_{j=1}^{n} \hat{f}(v_j). \tag{2}$$

It is obvious that $\hat{f}_{\langle \mathbf{v}, \mathbf{X} \rangle}(1) = 0$ if $\mathbf{X} \sim U(\mathbb{Z}_q^n)$ and $\mathbf{v} \neq \mathbf{0}$ since

$$\sum_{j=0}^{q-1} e^{\frac{-2\pi i j k}{q}} = 0.$$

However, the $\hat{f}_{\langle \mathbf{v}, \mathbf{X} \rangle}(1) \neq 0$ may hold for a suitable vector $\mathbf{v}$ and $\mathbf{x}_i \sim \chi$ that χ is other error distribution (i.e., $\hat{f}(v_j) \neq 0$ for all components of $\mathbf{v}$). A positive lower bound of $|\hat{f}_{\langle \mathbf{v}, \mathbf{X} \rangle}(1)|$ can be regarded as a distinguishing advantage. Therefore, as is shown in (2), the distinguish advantage corresponding to the vector $\mathbf{v}$ in the dual lattice is not only related to the length, but also the components.

The Fourier transform of practical error distributions is calculated in [11].

Theorem 4 ([11]).

- *For the variable X following approximate rounded Gaussian distribution $f(x)$, we have*

$$\left| \widehat{f}(k) - \widehat{\Psi_{s,q}}(k) \right| \leq 2e^{\frac{-\pi R^2}{s^2}}$$

where $k \in \mathbb{Z}_q$ and

$$\frac{2}{\pi} e^{-\frac{\pi s^2 k^2}{q^2}} \leqslant \widehat{\Psi_{s,q}}(k) \leqslant 2e^{-\frac{\pi s^2 k^2}{q^2}}.$$

- *For the variable X following central binomial distribution $B(h)$, let $k \in \mathbb{Z}_q$,* *we have*

$$\hat{f}(k) = \cos^{2h}(\frac{\pi k}{q}).$$

- *For the variable $X \sim U[-a, -a+1, \cdots, 0, 1, \cdots, b]$, we have*

$$|\hat{f}(k)|^2 = \frac{1 - \cos\frac{2\pi(a+b+1)k}{q}}{(a+b+1)^2(1-\cos\frac{2\pi k}{q})}$$

where $k = 1, 2, \cdots, \lfloor \frac{q}{2} \rfloor$.

It is shown that the component of the vector has influence on the complexity of distinguish. On the one hand, $\epsilon(\|\mathbf{v}\|) = e^{-\frac{\pi s^2\|\mathbf{v}\|^2}{q^2}}$ is used currently in estimating the distinguish advantage whatever the concrete error distribution is. On the other hand, the estimation under Fourier transform $\prod_{j=1}^{n} \hat{f}(v_j)$ reveals the difference of error distribution. A natural question is to make a comparison between $\epsilon(\|\mathbf{v}\|)$ and $\prod_{j=1}^{n} \hat{f}(v_j)$, therefore, we define local width $s(k)$.

Definition 16 (Local Width)**.** *For a given random variable X over $\mathbb{Z}_q$ and its probabilistic function $f(x)$, if $1 \leqslant k \leqslant \lfloor \frac{q}{2} \rfloor$ and $\hat{f}(k) \neq 0$, the local width $s(k)$ is defined to be*

$$s(k) = \frac{q}{k}\sqrt{\frac{-\ln|\hat{f}(k)|}{\pi}}.$$

The local width of practical error distributions is calculated and comparison between local width and given width is estimated in [11].

Theorem 5 ([11])**.** *1. For the central binormal distribution $B(h)$, we have*

$$s(k)^2 \geq s^2 + 2\pi h \left(\frac{k\pi}{q}\right)^2 \left(\frac{1}{12} + \frac{1}{45}(\frac{k\pi}{q})^2 + \frac{17}{2520}(\frac{k\pi}{q})^4\right),$$

for $k = 1, 2, \cdots, \lfloor \frac{q}{2} \rfloor - 1$ where $s = \sqrt{h\pi}$.

2. For the bounded uniform distribution $U[-a, -a+1, \cdots, 0, \cdots, b]$ with $a+b \geq 7$, we have

$$s(k)^2 + \frac{5(16qk - 3q^2)}{8k^2} < s^2,$$

for $k \in \{1, 2, \cdots, \lfloor q/2 \rfloor\}$ *with exceptions that* $k \geq \frac{3q}{16\pi}$ *and for any integer* ℓ, $|\frac{dk\pi}{q} - \ell\pi| \geq \frac{\pi}{24}$*. Here* $s = \sqrt{\pi\frac{(a+b+1)^2-1}{6}}$.

As for central binomial distribution, the lower bound of local width is the given width, therefore taking $\epsilon = e^{-\pi s^2 l^2/q^2}$ as the distinguish advantage is conservative. However, as for bounded uniform distribution, the local width is lower than given width in some intervals, which shows that the distinguish advantage corresponding to certain vectors is bigger than ϵ. It is noted that for the approximate rounded Gaussian distribution, the above theorem does not give the theoretical relationship between local width and given width as the comparison would closely related to the sampling method, truncation and other factors. Moreover, we can analyze the specific distribution directly. Taking the scheme parameters of the above three distributions as an example, the characteristics of the local width are given in the following.

- The error in Frodo scheme follows approximated rounded Gaussian distribution. Take one of the given parameters as example, that is

  $$(n, q, \sigma) = (640, 32768, 2.8),$$

  and the probability function is shown in Table 12.

Table 12: The Error Distribution in Frodo

standard deviation	0	±1	±2	±3	±4	±5	±6	±7	±8	±9	±10	±11	±12
2.8	$\frac{9288}{65536}$	$\frac{8720}{65536}$	$\frac{7216}{65536}$	$\frac{5264}{65536}$	$\frac{3384}{65536}$	$\frac{1918}{65536}$	$\frac{958}{65536}$	$\frac{422}{65536}$	$\frac{164}{65536}$	$\frac{56}{65536}$	$\frac{17}{65536}$	$\frac{4}{65536}$	$\frac{1}{65536}$

  The width is $s_0 = \sigma\sqrt{2\pi} = 7.02$. The comparison between local width $s(k)$ and s_0 is shown in Figure 4.

  As is shown in Figure 4, the local width is lower than s_0 when $k > q/3$. Therefore vectors of length $\ell > q/3$ in the dual lattice have different distinguish advantage. For example, for a vector $\mathbf{v}$ of length $q/2$, if it has a

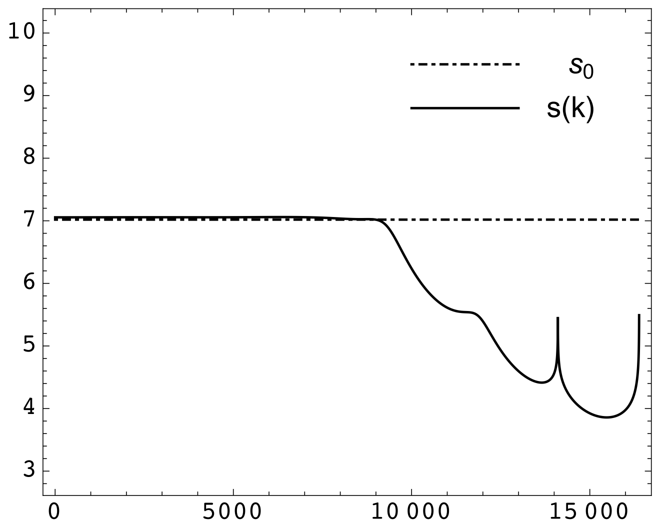

Figure 4: Comparison between local width $s(k)$ and given width s_0 in Frodo. The abscissa is the value of k in local width and the ordinate is the value of local width and given width.

component bigger than $q/3$, the lower bound of its distinguish advantage is

$$e^{-\pi(\frac{4}{9}s(\frac{q}{3})^2+\frac{5}{9}s_0^2)\|\mathbf{v}\|^2/q^2} \approx e^{-\pi(\frac{4*5.63^2}{9}+\frac{5*7.02^2}{9})\|v\|^2/q^2} \approx e^{-\pi 6.43^2\|v\|^2/q^2}$$

which is obviously bigger than $e^{-\pi s_0^2\|v\|^2/q^2}$. Furthermore, if the components of vector $\mathbf{v}$ are all smaller than $q/3$, its corresponding distinguish advantage is close to $e^{-\pi s_0^2\|v\|^2/q^2}$.

- The NewHope scheme is based on RLWE problem and its error follows central binomial distribution. We take one of its parameters as example, that is

$$(n, q, \sigma) = (512, 12289, 2),$$

where the error $e \sim B(8)$. The comparison between local width $s(k)$ and $s_0 = 2\sqrt{2\pi} \approx 5.01$ is shown in Figure 5 .

It is shown that $s(k)$ is always bigger than s_0 for any $k \leqslant q/2$. Furthermore, the value $s(k) - s_0$ increases with k increasing which is consistent with the

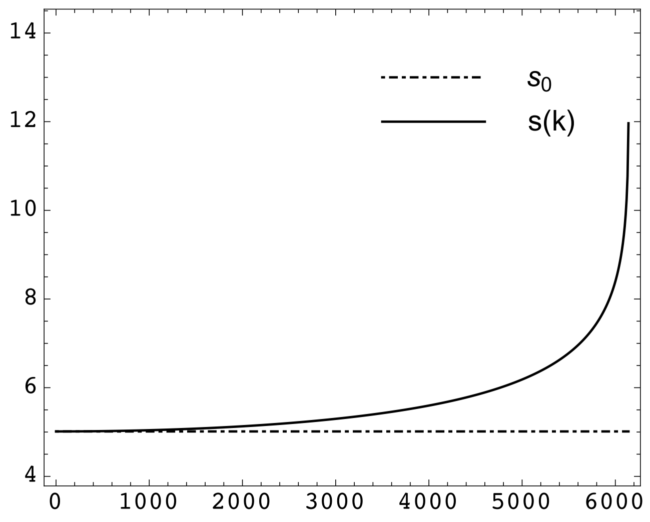

Figure 5: Comparison between local width $s(k)$ and given width s_0 in NewHope. The abscissa is the value of k in local width and the ordinate is the value of local width and given width.

result in Theorem 5. Moreover, it shows that taking $e^{-\pi s_0^2\|\mathbf{v}\|^2/q^2}$ as the estimation of distinguish advantage is conservative.

- The Saber scheme is based on MLWR problem. The (M)LWR problem is usually treated as LWE problem with error following bounded uniform distribution. Take one of parameters in Saber as example, that is

  $$(n, q, \sigma) = (768, 8192, 2.29),$$

  where $e \sim U[-3, -2, -1, 0, 1, 2, 3, 4]$. The comparison between local width $s(k)$ and $s_0 = 2\sqrt{2\pi} \approx 5.01$ is shown in Figure 6 .

  As is shown in Figure 6, there are gaps between $s(k)$ and s_0 for certain $k's$, e.g. $s(k)$ is strictly smaller than $s(0)$ when $k \geq q/5$. The property is consistent with the result in Theorem 5. Similarly, it shows that some vectors in the dual lattice correspond to bigger distinguish advantage when compared the current estimation.

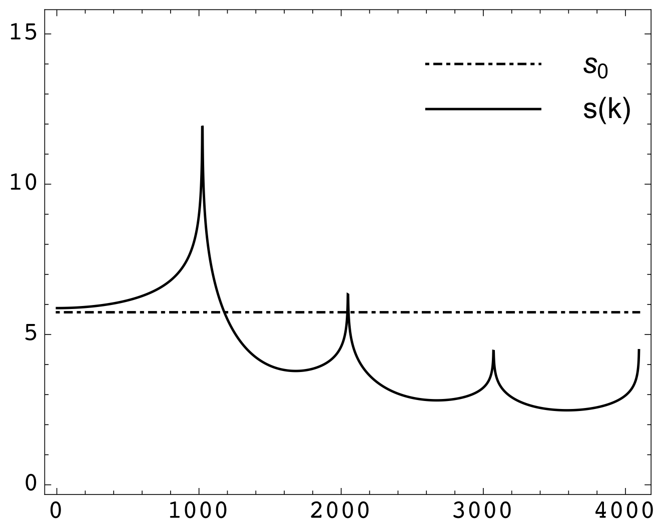

Figure 6: Comparison between local width $s(k)$ and given width s_0 in Saber. The abscissa is the value of k in local width and the ordinate is the value of local width and given width.

B Mixed Sampler

According to the definition of mixed distribution, we can prove that the local width of mixed sampler is always bigger than the given width.

Theorem 6 ([11])**.** *Let q be an integer and the variable $X \sim \chi(k_1, k_2)$, if $k_1 \geqslant k_2$, we have*

$$s(j) \geqslant s_0$$

for $j = 1, 2, \cdots, \lfloor q/2 \rfloor$ where $s_0 = \sqrt{2\pi D[X]} = \sqrt{(k_1 + 4k_2/3)\pi}$.

Mixed sampling can be implemented efficiently by taking the central binomial sampler and the uniform sampler on $\{-1, 0, 1\}$ as underlying algorithms. The sampling algorithm is listed as follows.

Mixed sampling algorithm

Input: The parameter (k_1, k_2).
Next page

Mixed sampling algorithm

```
Output: The value of variable following the distribution $\chi(k_1, k_2)$.
1: To sample from a central binomial distribution, first set a random number generated by a 2-bit random source and output an integer $a$ that subjects to a central binomial distribution. For example, when the random input is 00\01\10\11 respectively, the output is $-1$\0\0\1.
2: Repeat the central binomial distribution sampling $k_1$ times, then calculate the sum of $k_1$ values. That is, let the $i$-th output is $a_i$ for $i = 1, \cdots, k_1$, then calculate $A = a_1 + a_2 + \cdots + a_{k_1}$.
3: Take a random number generated by $f = \lceil \log_2(3^{k_2}) \rceil$ bits random source. If the value of the random number in binary is greater than $3^{k_2}$, then enter a random number generated by the $f$ bits random source again until the value is not greater than $3^{k_2}$. Let the random number be expressed as a $k_2$ ternary string, then count the number of $0, 2$ which is denoted respectively as $b_0, b_2$, output $B = b_2 - b_0$;
4: Output the value $S = A + B \bmod q$.
```

The efficiency of mixed sampling algorithm is proved.

Theorem 7 ([11]). *The mixed sampling algorithm outputs a sample distributed as $\chi(k_1, k_2)$ correctly and the expectation of bits used to output a sample is*

$$2k_1 + \frac{\lceil k_2 \log_2 3 \rceil 2^{\lceil k_2 \log_2 3 \rceil}}{3^{k_2}}.$$

The comparison of variance, local width and efficiency between different samplers is shown in Table 14. It should be noted that there is no close lower bound of local width for bounded uniform sampling. Therefore we make comparison of efficiency and variance among other samplers (that is, we give the comparison among mixed sampling, central binormal sampling and discrete Gaussian sampling). The comparison is shown in Figure 7.

Table 14: Comparison among different samplings.

sampling algorithm	bits	variance	lower bound of local width
discrete Gaussian	$(\pi s^2 \log_2 e)/4$	$s^2/(2\pi)$	s
mixed sampling	$2k_1 + f'2^{f'}/3^{k_2}$	$k_1/2 + 2k_2/3$	$\sqrt{(k_1 + 4k_2/3)\pi}$ *
central binomial	$2k$	$k/2$	$\sqrt{k\pi}$
bounded uniform	$f2^f/3^k$	$2k/3$	no lower bound

$f = \lceil \log_2(3^k) \rceil, f' = \lceil \log_2(3^{k_2}) \rceil$.

C Algebraic Attack

Algebraic attack was first proposed by Arora and Ge [4], it is an alternative approach to solving Search-LWE by setting up a system of noise-free non-linear polynomials of which the secret s is a root. This approach can find s directly.

In particular, the algorithm proceeds by assuming that the error always falls in the rang $[-t, t]$ ($t \in \mathbb{Z},\ d = 2t + 1 < q$), then the polynomials are constructed from the observation that the error is always a root of the polynomial $P(x) = x \prod_{i=1}^{t} (x + i)(x - i)$. Moreover, we know the secret s is a root of $P(a \cdot x - b)$.

In the Arora-Ge algorithm [4], the system of nonlinear equations constructed delete??this way is solved by replacing each monomial with a new variable and solving the resulting linear system. This can be improved by using *Gröbner* basis technique [1], which needs less equations but more operations.

Theorem 8 (Theorem *5* in [1])**.** *Let* (n, q, σ) *be parameters of an* LWE *instance,* $2 \le \omega < 3$ *denote the linear algebra constant and let* $D_{AG} = 8\sigma^2 \log n + 1$*. If* $D_{AG} \in o(n)$ *then the Arora-Ge algorithm solves Search-LWE in time complexity*

$$O(2^{\omega \cdot D_{AG} \log \frac{n}{D_{AG}}} \cdot \sigma q \log q) = O(2^{8\omega\sigma^2 \log n (\log n - \log(8\sigma^2 \log n))} \cdot ploy(n)),$$

and memory complexity

$$O(2^{2 \cdot D_{AG} \log \frac{n}{D_{AG}}} \cdot \sigma q \log q) = O(2^{16\sigma^2 \log n (\log n - \log(8\sigma^2 \log n))} \cdot ploy(n));$$

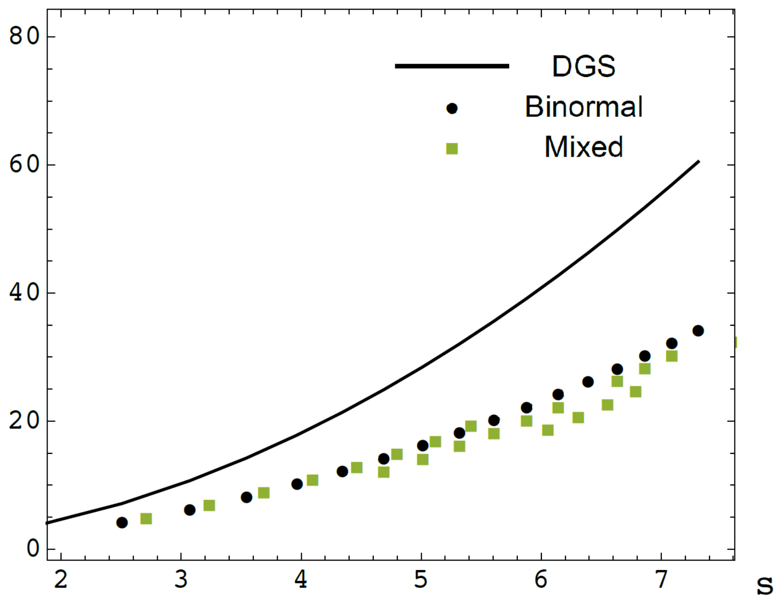

Figure 7: Efficiency comparison of sampling algorithms

*'DGS' stands for discrete Gaussian sampling, 'Binormal' stands for central binormal sampling and 'Mixed' stands for discrete Gaussian sampling.
*The **s**-axis denotes the sampler's width and longitudinal axis is the bits in the sampling algorithm. The square symbol denotes the width and bits under proper choice of parameters (k_1, k_2) in mixed sampling.
*Let y be the bits of sampling, then

$$y = \begin{cases} 1.1331s^2, \text{ when it is discrete Gaussian sampling;} \\ 0.6366s^2, \text{ when it is binormal sampling;} \\ 0.4442s^2 + 3.9269s - 23.7935, \text{ when it is mixed sampling.} \end{cases}$$

If $n \in o(D_{AG})$ then the Arora-Ge algorithm solves Search-LWE in time complexity

$$O(2^{\omega \cdot n \log \frac{D_{AG}}{n}} \cdot \sigma q \log q) = O(2^{\omega n \log(8\sigma^2 \log n) - \omega n \log n} \cdot ploy(n)),$$

and memory complexity

$$O(2^{2 \cdot n \log \frac{D_{AG}}{n}} \cdot \sigma q \log q) = O(2^{2n \log(8\sigma^2 \log n) - 2n \log n} \cdot ploy(n)).$$

Taking $\sigma = \sqrt{n}$ as an example, we give the asymptotic complexity of the two algorithms. The asymptotic complexity of the basic Arora-Ge algorithm is $O(2^{(2+\epsilon)\omega n\log\log n})$ [3]. Combined with *Gröbner* base, the time complexity is $O(2^{2.35\omega n+1.13n})$, the storage complexity is $O(2^{5.85n})$ and the number of samples needed is $m = \exp(\pi n/4)$ [3].

D BKW Attack

The BKW algorithm was firstly proposed by Blum, Kalai and Wasserman [8] and its core idea is to select a few samples by generalized birthday attack so as to make the sum of selected samples being zero vector. Then it can be used as the distinguisher of LWE instances.

In more detail, BKW constructs the $\mathbf{v}_i$ as follows. Given samples $\mathbf{A} \in M_{m\times n}$, BKW splits the n components into a blocks each of width $b = n/a$. There are a stages of the algorithm in which the algorithm creates tables by searching for collisions in the appropriate b coefficients of $\mathbf{A}$. In the i-th stage, the samples that its elements in $i - th$ blocks are zeros, are obtained by collisions. Assuming that the initial number of samples is $2^a q^b$, at the end of the algorithm in the optimal case, we can obtain q^b short vectors with the length of $\sqrt{2^a}$.

Next, the BKW attack uses the short vector to distinguish LWE distribution from uniform distribution, which is similar with the dual attack introduced in Section 5.3.2. The advantage of each short vector is $\varepsilon = \exp(-2\pi^2 \cdot 2^a(\frac{\sigma}{q})^2)$, then BKW attack requires about $1/\varepsilon^2$ short vectors to enlarge the success rate. Because BKZ algorithm can generate q^b vectors, the condition of $1/(\exp(-2\pi^2 \cdot 2^a(\frac{\sigma}{q})^2))^2 \leq q^{n/a}$ should be satisfied by carefully selecting the parameter a.

Using BKW algorithm to solve LWE problem is firstly proposed by Albrecht et al. on PKC 2014[2]. In the next year, the two algorithms, i.e., [16] and [19], are independently proposed to improve BKW algorithm. Here, we use the expression in [16] which called the improved algorithm as coded-BKW algorithm. The other [15] combines the coded-BKW algorithm and sieve which can get the optimal time complexity.

Theorem 9. [15] *Let (n, q, σ) be the parameters of LWE instance, and $q = n^{c_q}, \sigma = n^{c_s}$。The time and storage complexity of coded-BKW combined with sieving both are $2^{(c+o(1))n}$, where*

$$c = \frac{1}{1 - e^{-\lambda(1+2(c_q-c_s))/c_s}},$$

$\lambda = 0.292$ *in classical case and* $\lambda = 0.265$ *in quantum case.*

Theorem 10. [15] *If $c > \lambda$ and $\frac{c_s}{\lambda} \ln \frac{c_q}{c_s} < 2(c_q - c_s) + 1$, the expected cost of the coded-BKW algorithm using BKW as preprocess to distinguish LWE distribution from uniform distribution is* $2^{(c+o(1))n}$, *where*

$$c = (\lambda^{-1}(1 - \frac{c_s}{c_q}) + c_q^{-1}(2(c_q - c_s) + 1 - \frac{c_s}{\lambda} \ln \frac{c_q}{c_s}))^{-1}$$

$\lambda = 0.292$ *in classical case and* $\lambda = 0.265$ *in quantum case.*

Using the Regev's LWE instance that $q = n^2$ and $\sigma = n^{1.5}/(\sqrt{2\pi} \log_2^2 n)$ as example, the complexity of BKW algorithm according to Theorem 10 is $2^{0.895n}$ which is the best result at present.

As for our schemes, (1) in the efficient parameter scheme, the secret will be changed regularly and the adversary cannot get the number of samples needed for algebraic attack, so BKW attack cannot be implemented. (2) In the quantum security scheme, according to theorem 6, we give the attack complexity of BKW algorithm with sieving on different parameters, as shown in Table 6.

E Security against BKW Attack and Algebra Attack

Table 15: Security of SCloud in high security & long message mode against BKW with sieving

	q640	q896	q1216
security level(bit)	128	192	256
(n, q, σ)	$(640, 2^{16}, 4.10)$	$(896, 2^{17}, 4.83)$	$(1216, 2^{18}, 5.85)$
classical	2^{145}	2^{206}	2^{284}
quantum	2^{138}	2^{196}	2^{271}

Table 16: Security of SCloud in high security & long message mode against algebra attack

	q640	q896	q1216
security level(bit)	128	192	256
(n, q, σ)	$(640, 2^{16}, 4.1)$	$(896, 2^{17}, 4.83)$	$(1216, 2^{18}, 5.85)$
time complexity	$O(2^{620.8\omega})$	$O(2^{923.4\omega})$	$O(2^{1467\omega})$
space complexity	$O(2^{1242})$	$O(2^{1847})$	$O(2^{2933})$

Table 17: Security of SCloud in high security & short message mode against BKW with sieving

	qg640	qg896	qg1216
security level(bit)	128	192	256
(n, q, σ)	$(640, 2^{16}, 4.10)$	$(896, 2^{17}, 4.83)$	$(1216, 2^{18}, 5.85)$
classical	2^{145}	2^{206}	2^{284}
quantum	2^{138}	2^{196}	2^{271}

Table 18: Security of SCloud in high security & short message mode against algebra attack

	qg640	qg896	qg1216
security level(bit)	128	192	256
(n, q, σ)	$(640, 2^{16}, 4.1)$	$(896, 2^{17}, 4.83)$	$(1216, 2^{18}, 5.85)$
time complexity	$O(2^{620.8\omega})$	$O(2^{923.4\omega})$	$O(2^{1467\omega})$
space complexity	$O(2^{1242})$	$O(2^{1847})$	$O(2^{2933})$