Scloud$^+$: a Lightweight LWE-based KEM without Ring/Module Structure

Anyu Wang[1,5,6], Zhongxiang Zheng[2(✉)], Chunhuan Zhao[3], Zhiyuan Qiu[4], Guang Zeng[3], and Xiaoyun Wang[1,4,5,6,7(✉)]

[1] Institute for Advanced Study, BNRist, Tsinghua University, Beijing, China
{anyuwang,xiaoyunwang}@tsinghua.edu.cn,
[2] School of Computer and Cyber Sciences, Communication University of China, Beijing, China
zhengzx@cuc.edu.cn
[3] Shield Lab, Huawei Technology, Beijing, China
{zhaochunhuan,zengguang13}@huawei.com,
[4] Shandong Institute of Blockchain, Jinan, China
qiuzhiyuan@sdibc.cn
[5] Zhongguancun Laboratory, Beijing, China
[6] National Financial Cryptography Research Center, Beijing, China
[7] Key Laboratory of Cryptologic Technology and Information Security (Ministry of Education), School of Cyber Science and Technology, Shandong University, Qingdao, China

Abstract. We propose Scloud$^+$, a lattice-based key encapsulation mechanism (KEM) scheme. The design of Scloud$^+$ is informed by the following two aspects. Firstly, Scloud$^+$ is based on the hardness of *algebraic-structure-free* lattice problems, which avoids potential attacks brought by the algebraic structures. Secondly, Scloud$^+$ provides sets of *light weight parameters*, which greatly reduce the complexity of computation and communication complexity while maintaining the required level of security.

Keywords: post-quantum cryptography · key encapsulation mechanism · learning with errors · lattice code · Barnes-Wall lattice

1 Introduction

Shor's quantum algorithm [1] makes the migration to post-quantum public key cryptography an inevitable. Amongst the post-quantum public key schemes, those based on the learning with errors (LWE) problem are prevalent. The LWE problem was firstly studied by Regev in 2005 [2], which roughly requires to solve a noisy linear equation system modulo a known positive integer. Regev proved that the LWE problem is at least as hard as the approximate shortest vector problem (SVP) and the shortest independent vectors problem (SIVP) on random lattices, which are believed still to be hard in quantum world.

Since the first LWE-based public encryption algorithm proposed by Regev [2], various schemes have been developed based on the hardness of LWE. According to whether adopting algebraic structure in the LWE problem, these schemes can be divided into two classes. The first class bases its security on the hardness of the LWE problem without introducing additional algebraic structures, which includes FrodoKEM [3]. The second class of schemes are constructed based on some variants of the LWE problem with algebraic structures, e.g., the Ring-LWE problem [4,5] and the Module-LWE [6]. These schemes include CRYSTALS-Kyber [7], Saber [8], LAC [9], Aigis [10], etc.

The biggest benefit of introducing algebraic structure is making it possible to construct LWE-based public key schemes that are 'compact', i.e., efficient with respect to the computation and communication complexity. However, the algebraic structure also makes it unlikely to reduce the hardness of the Ring-LWE problem and the Module-LWE to the hard problems on (algebraic-unstructured) random lattices, such as the approximate SVP and the SIVP. Alternatively, it is known that the variant LWE problems can be reduced to the problems on with algebraic structured lattices. Specifically, the Ring-LWE problem is proved at least as hard as the approximate Ideal-SVP [4], and the Module-LWE problem is roved at least as hard as the approximate Module-SVP [6]. However, different from the approximate SVP and the SIVP, the hardness of the approximate Ideal-SVP and the approximate Module-SVP under quantum computing remain debatable. In fact, several efficient quantum algorithms for the approximate Ideal-SVP are discovered recently. In 2016, Cramer et al. proved that the approximate Ideal-SVP for specific cyclotomic fields with approximation factor $2^{\tilde{O}(\sqrt{n})}$ can be solve in quantum polynomial time [11], while the best known algorithm for the approximate SVP with the same approximation factor is still sub-exponential [12]. This result has been extended to general cyclotomic fields [13,14,15,16], and arbitrary number fields [17,18]. Although it seems unlikely to extend these approaches to directly tackle the approximate Module-SVP and the Ring-LWE/Module-LWE problems, the impact of the algebraic structure on the security is still far from clear.

1.1 Design Rationale

Scloud$^+$ aims to provide a key encapsulation mechanism (KEM) scheme based on the hardness of the *algebraic-unstructured* LWE problem. Notably, FrodoKEM [3] has already provided such a solution. This choice enables resistance to potential attacks against algebraic structures but also limits efficiency. To optimize communication and computation efficiency, Scloud$^+$ leverages carefully selected secret/error distributions and finely designed error-correcting codes, offering sets of *lightweight parameters*. These techniques significantly enhance efficiency while maintaining the required level of security.

2 Preliminaries

2.1 Notations

Vectors and Matrices. Vectors are denoted by bold lower-case letters, e.g., $\mathbf{v}$, and matrices are denoted by bold upper-case letters, e.g., $\mathbf{A}$. The i-th entry of an n dimensional vector $\mathbf{v}$ is denoted by $\mathbf{v}[i], 0 \leq i < n$. The (i, j)-th entry of an $m \times n$ matrix $\mathbf{A}$ is denoted by $\mathbf{A}[i, j], 0 \leq i < m, 0 \leq j < n$, and the i-th row (or the i-th column) of $\mathbf{A}$ is denoted by $\mathbf{A}[i, \cdot]$ (or $\mathbf{A}[\cdot, j]$). For a vector $\mathbf{v}$, let $w_H(\mathbf{v})$ denote the hamming weight of $\mathbf{v}$, i.e, $w_H(\mathbf{v})$ = the number of nonzero elements in $\mathbf{v}[i]'s, 0 \leq i < n$. For a real vector $\mathbf{v} \in \mathbb{R}^n$, let $\|\mathbf{v}\| = \sqrt{\sum_{i=0}^{n-1} \mathbf{v}[i]^2}$ denote its Euclidean norm. For two n-dimensional vectors $\mathbf{u}, \mathbf{v}$, let $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=0}^{n-1} \mathbf{u}[i] \cdot \mathbf{v}[i]$ denote their inner product.

We use $V^{(n,h)}$ to denote the set of n dimensional vectors which contains exactly $(n - 2h)$ '0's, h '1's and h '-1's. Let $H^{(m,n,h)}$ and $L^{(m,n,h)}$ be two sets of $m \times n$ matrices such that $H^{(m,n,h)} = \{\mathbf{A} : \mathbf{A}[i, \cdot] \in V^{(n,h)} \text{ for } 0 \leq i < m\}$, and let $L^{(m,n,h)} = \{\mathbf{B} : \mathbf{B}[\cdot, i] \in V^{(m,h)} \text{ for } 0 \leq i < n\}$.

For $x \in \mathbb{R}$, we use $\lfloor x \rfloor$ to denote the largest integer less than or equal to x, and use $\lfloor x \rceil = \lfloor x + 1/2 \rfloor$ to denote the integer closest to x.

Distributions and Sampling Functions. For a distribution χ, let $x \hookleftarrow \chi$ denote sampling an x according to χ. Let $U(q)$ denote a uniform discrete distribution on $[0, 1, \cdots, q - 1]$. We also define the other two distributions here, central binomial distribution and fixed Hamming distribution.

Central binomial distribution. Let $\rho(k)$ denote the centered binomial distribution with parameter k. For a random variable $X \hookleftarrow \rho(k)$, it can be written as $X = x_1 + x_2 + \cdots + x_k$ where x_i is the variable defined over $\{-1, 0, 1\}$ with $\texttt{Pr}[x_i = 0] = \frac{1}{2}$ and $\texttt{Pr}[x_i = 1] = \texttt{Pr}[x_i = -1] = \frac{1}{4}$.

Fixed Hamming Distribution. For a random variable X that follows a fixed hamming distribution with parameter h, denoted as $x \hookleftarrow \beta(h)$, is sampled with exactly $(n - 2h)$ '0's, h '1's and h '-1's.

2.2 LWE and LWR Problems

An n dimensional full rank lattice L is a discrete additive group in $\mathbb{R}^n$. For a lattice L with the basis $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n]$, the vectors in L can be represented as the integer combinations of $\mathbf{B}$, i.e.

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

For the lattice L, we use $\lambda_1(L)$ denotes the length of shortest non-zero lattice vector.

One of the average-case problem related to lattice is the LWE problem that is proposed by Regev [2] and its security is based on the hardness of lattice computational problem. First we give the related definition here.

Definition 1 (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 $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e \; mod \; q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$.*

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 2 (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 3 (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.*

Variants of LWE problem are proposed successively, for example, the Ring-LWE, Module-LWE and the LWR problem[TODO ADD CITE]. The LWR problem can be seen as the derandomized version of LWE problem and its definition is as follows.

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

Similarly, the LWR problem also has the search and decision version. It is easily seen that the noise of LWR is deterministic since it can be written as

$$\lceil \frac{p}{q}(\langle \mathbf{a}, \mathbf{s} \rangle \; mod \; q \rfloor) = \frac{p}{q}(\langle \mathbf{a}, \mathbf{s} \rangle + e \; mod \; q)$$

where e is determined by the reminder of $\langle \mathbf{a}, \mathbf{s} \rangle$ and can be seen as the uniformly distribution over the interval $(-\frac{p}{2q}, \frac{p}{2q}]$.

2.3 Barnes-Wall Lattice

The Barnes-Wall lattice [19] is a family of lattice and has been well studied in coding theory and mathematics. It is well known for its packing property especially for the lower dimensional BW lattice. For $n = 2, 4, 8, 16$, the lattices are known as $\mathbb{Z}^2$, D_4, E_8, L_{16} lattice which are of densest packing in its dimension respectively.

Let $\phi = 1+i$, the BW lattice of dimension $n = 2^k$ in $\mathbb{C}^n$ is a lattice generated by the rows of the matrix

$$W_n = \begin{bmatrix} 1 & 1 \\ 0 & \phi \end{bmatrix}^{\otimes k} \in \mathbb{C}^{n \times n}$$

Equivalently, it can be defined iteratively as follows.

Definition 5 (Barnes-Wall lattice [20]). *For any positive integer $n = 2^k \geq 4$, the n-th Barnes-Wall lattice BW_n is defined as*

$$BW_n = \{[\mathbf{u}, \mathbf{u} + \phi\mathbf{v}] : \mathbf{u}, \mathbf{v} \in BW_{n/2}\}$$

where $BW_2 = \mathbb{Z}[i]$.

According to the definition, for the BW_n lattice vectors, it can be written as the form

$$\begin{bmatrix} 1 & 0 \\ 1 & \phi \end{bmatrix} \cdot \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}$$

then for the shortest vectors in BW lattice, it has the following property.

Lemma 1 *For any positive integer $n = 2^k$ where $k > 1$, there is $\lambda_1(BW_{2n}) = \sqrt{2}\lambda_1(BW_n)$.*

As $\lambda_1(BW_2) = 1$, we have $\lambda_1(BW_n) = \sqrt{\frac{n}{2}}$. For the packing radius ρ of BW lattice, there is

$$\rho(BW_n) = \frac{\lambda_1(BW_n)}{2} = \frac{\sqrt{n/2}}{2}.$$

For the determination of BW_n, it has $\det(BW_n) = 2^{\frac{n}{4}}(\det(BW_{\frac{n}{2}}))^2$, and thus

$$\det(BW_n) = \left(\frac{n}{2}\right)^{\frac{n}{4}}.$$

Several algorithms have been proposed to decode BW lattices, which can be broadly categorized into two types. The first category focuses on Maximum Likelihood Decoding (MLD). A notable example is the algorithm proposed by Forney in 1988, which utilizes the trellis representation of BW_n [21]. However, this algorithm becomes computationally infeasible for $n > 32$ [22]. The second category, initiated by Micciancio and Nicolosi in 2008, centers on Bounded Distance Decoding (BDD) [23]. This approach aims to find the unique lattice point u in BW_n such that $\mathrm{dist}(y, BW_n) = \mathrm{dist}(y, u)$ for any given point y where $\mathrm{dist}(y, BW_n) < \rho(BW_n)$. Additionally, the list decoding of BW lattices has been explored by Grigorescu et al. in 2012 [20].

3 Message Encoding and Decoding using BW Lattice

In this section, we introduce our message encoding and decoding method using the BW lattice.

3.1 Encoding in the BW Lattice Code

Let $\mathbf{m} \in \{0,1\}^d$ denote the d-bit message vector. We consider encoding $\mathbf{m}$ into the lattice vector in $BW_n \cap \mathbb{Z}_{2^r}^n$ for a modulus parameter r. Let $B \in \mathbb{Z}^{n\times n}$ denote the basis matrix of BW_n. The encoding of a lattice code is typically performed as follows:

$$\mathbf{m} \in \{0,1\}^d \xrightarrow{\text{Step 0}} \mathbf{x} \in \mathbb{Z}^n \xrightarrow{\text{Step 1}} \mathbf{y} = \mathbf{B}\mathbf{x} \mod 2^r \in BW_n.$$

Note that the above process needs to be injective to ensure that decoding is possible. Clearly, the message space must satisfy

$$d \leq \log_2 \left(\frac{2^{rn}}{\det(B)} \right).$$

Additionally, B should be selected to ensure the injective property. For large dimensions n, storing a selected B as a "magic matrix" can hinder readability and optimization in implementation. To address this, we propose a new iterative encoding procedure which is more natural to implement.

An Iterative Encoding Method. Recall that one vector in BW_n is always combined with two vectors in $BW_{n/2}$, i.e.

$$BW_n = \{[\mathbf{u}, \mathbf{u} + \phi\mathbf{v}] : \mathbf{u}, \mathbf{v} \in BW_{n/2}\}$$

Take the BW_{16} as an example, for a vector $y \in BW_{16}$, it could calculated by the two vectors in BW_8, and for the vectors in BW_8, they could be written with vectors in BW_4 and so on, which is shown in the Figure 1. The iterative method is given in Algorithm 1. Instead of calculate the matrix-vector multhiplication, our iterative method avoid the store of the specific basis B.

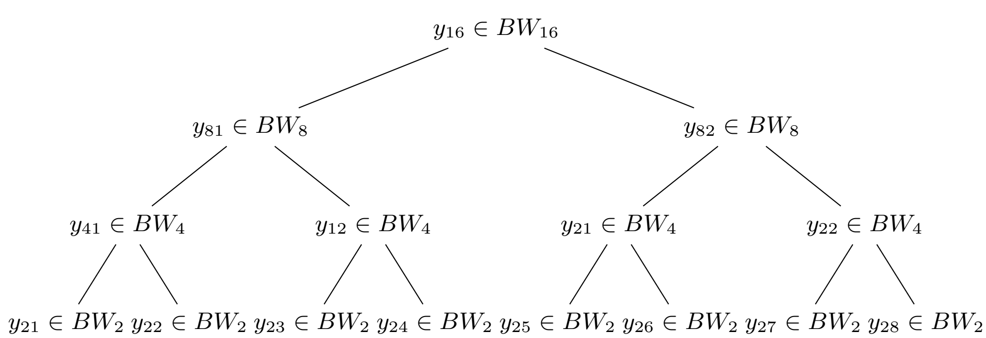

Fig. 1. The construction structure of vectors in BW_{16}

In order to be compatible with the space in $B \in \mathbb{Z}^{n\times n}$, we need to deal with the zero space in the mapping. Here we give our observation.

Algorithm 1: Iterative Encoding($\mathbf{x}$, $n = 2^k$)

Input: $x \in \mathbb{Z}^n$

Output: $y \in BW_n$

1: $(y_1, y_2, \cdots, y_{n/2}) := (x_1 + ix_2, \cdots, x_{n/2-1} + ix_{n/2}) \hookleftarrow (x_1, x_2, \cdots, x_n)$

2: For $i = 1, 2, \cdots, k-1$

3: $(y_1, y_2, \cdots, y_{n/2}) \hookleftarrow (y_1, y_2, \cdots, y_{2^i}, (y_1, y_2, \cdots, y_{2^i}) + \phi(y_{2^i+1}, y_{2^i+2}, \cdots, y_{2^{i+1}}), \cdots, y_{2^{k-2i}}, y_{2^{k-2i}+1}, \cdots, y_{2^{k-i}}, (y_{2^{k-2i}}, y_{2^{k-2i}+1}, \cdots, y_{2^{k-i}}) + \phi(y_{2^{k-i}+1}, y_{2^{k-i}+2}, \cdots, y_n))$

4: **return** $y \in BW_n$

Lemma 2 *Let $\mathbb{K} = \mathbb{Z}[i] = \{a + bi : a, b \in \mathbb{Z}\}$, $n = 2^k$. Let $\mathbf{B}$ denote the basis matrix of BW_n corresponding to iterative encoding process. Assume that $\phi^{k-1} | 2^r$, Let $\Phi = (2^r, \frac{2^r}{\phi}, \frac{2^r}{\phi}, \frac{2^r}{\phi^2}, \cdots, \frac{2^r}{\phi^{k-1}}) \in \mathbb{K}^{n/2}$ where $\Phi[i] = 2^r / \phi^{\text{the hamming weight of } i}$, there exists an bijective map from $\mathbf{x} \in \mathbb{Z}^n/\Phi$ to $f(\mathbf{x}) = \mathbf{B}\mathbf{x} \mod 2^r$.*

Proof. For the map

$$f : \mathbf{x} \in \mathbb{Z}^n/\Phi \longrightarrow \mathbf{B}\mathbf{x} \mod 2^r,$$

Recall that for the matrix $\mathbb{B}$, it could written into $k-1$ matrix multiplication, that is

$$\mathbf{B} = \mathbf{B_{k-1}} \mathbf{B_{k-2}} \cdots \mathbf{B_1},$$

where

$$\mathbf{B_{k-1}} = \begin{pmatrix} I_{2^{k-2} \times 2^{k-2}} & \mathbf{0} \\ I_{2^{k-2} \times 2^{k-2}} & \phi I_{2^{k-2} \times 2^{k-2}} \end{pmatrix} \in \mathbb{K}^{n/2 \times n/2},$$

$$\mathbf{B_{k-2}} = \left(\begin{array}{cc:cc} I_{2^{k-3} \times 2^{k-3}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ I_{2^{k-3} \times 2^{k-3}} & \phi I_{2^{k-3} \times 2^{k-3}} & \mathbf{0} & \mathbf{0} \\ \hdashline \mathbf{0} & \mathbf{0} & I_{2^{k-3} \times 2^{k-3}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & I_{2^{k-3} \times 2^{k-3}} & \phi I_{2^{k-3} \times 2^{k-3}} \end{array}\right) \in \mathbb{K}^{n/2 \times n/2},$$

$$\cdots\cdots$$

$$\mathbf{B_1} = \left(\begin{array}{cc:cc:c:cc} I_{1\times1} & \mathbf{0} & & & & & \\ I_{1\times1} & \phi I_{1\times1} & & & & & \\ \hdashline & & I_{1\times1} & \mathbf{0} & & & \\ & & I_{1\times1} & \phi I_{1\times1} & & & \\ \hdashline & & & & \ddots & & \\ & & & & \ddots & & \\ \hdashline & & & & & I_{1\times1} & \mathbf{0} \\ & & & & & I_{1\times1} & \phi I_{1\times1} \end{array}\right) \in \mathbb{K}^{n/2 \times n/2}.$$

Firstly we prove that the map is single mapping, we only need prove that for $\mathbf{x} \in \mathbb{Z}^n$, if there is $f(x) = 0$, i.e. $\mathbf{Bx} = 0 \mod 2^r$, then $\mathbf{x} \in < \Phi >$. Since the matrix $\mathbf{B}$ corresponds to the iterative encoding process, let $x = (x_1, x_2, \cdots, x_n)$ which can be written into $\mathbf{y}^{(0)} = (y_1, y_2, \cdots, y_{n/2}) \longleftarrow (x_1 + ix_2, \cdots, x_{n-1} + ix_n) \in \mathbb{K}^{n/2}$. So the map could be written as

$$\mathbf{Bx} = \mathbf{B_{k-1}B_{k-2}} \cdots \mathbf{B_1 y} \mod 2^r.$$

- According to the iterative definition, if

$$\mathbf{B_{k-1}B_{k-2}} \cdots \mathbf{B_1 y} = 0 \mod 2^r,$$

let

$$\mathbf{y}^{(i)} = \mathbf{B_i} \cdots \mathbf{B_1 y} \mod 2^r := \begin{pmatrix} y_1^i \\ y_2^i \\ \cdots \\ y_{n/4}^i \\ \hline y_{n/4+1}^i \\ y_{n/4+2}^i \\ \cdots \\ y_{n/2}^i \end{pmatrix} \in \mathbb{K}^{n/2},$$

then according to the structure of $\mathbf{B_{k-1}}$, we can get the property of $\mathbf{y}^{(k-2)}$ which is

$$\begin{pmatrix} y_1^{(k-2)} \\ y_2^{(k-2)} \\ \cdots \\ y_{n/4}^{(k-2)} \end{pmatrix} = 0 \mod 2^r, \quad \begin{pmatrix} y_{n/4+1}^{(k-2)} \\ y_{n/4+2}^{(k-2)} \\ \cdots \\ y_{n/2}^{(k-2)} \end{pmatrix} = 0 \mod 2^r/\phi,$$

- As the $\mathbf{y}^{(k-2)}$ is obtained from the multiplication of $\mathbf{B}_{k-2}$ and $\mathbf{y}^{(k-3)}$, then combined the structure of $\mathbf{B}_{k-2}$, there is

$$\begin{pmatrix} y_1^{(k-3)} \\ y_2^{(k-3)} \\ \cdots \\ y_{2^{(k-3)}}^{(k-3)} \end{pmatrix} = 0 \mod 2^r, \quad \begin{pmatrix} y_{2^{(k-3)}+1}^{(k-3)} \\ y_{2^{(k-3)}+2}^{(k-3)} \\ \cdots \\ y_{2^{(k-2)}}^{(k-3)} \end{pmatrix} = 0 \mod 2^r/\phi,$$

$$\begin{pmatrix} y_{2^{(k-2)}+1}^{(k-3)} \\ y_{2^{(k-2)}+2}^{(k-3)} \\ \cdots \\ y_{2^{(k-1)}}^{(k-3)} \end{pmatrix} = 0 \mod 2^r/\phi, \quad \begin{pmatrix} y_{2^{(k-1)}+1}^{(k-3)} \\ y_{2^{(k-1)}+2}^{(k-3)} \\ \cdots \\ y_{2^{(k)}}^{(k-3)} \end{pmatrix} = 0 \mod 2^r/\phi^2,$$

– Therefore it is easily to found that for $\mathbf{y}^{(0)}$, for $d = 1, 2, 3, \cdots, n/2$, there is

$$\mathbf{y}_d^{(0)} = 0 \mod 2^r/\phi^{\text{HammingWeightOf}(d-1)}, i.e. \mathbf{y}^{(0)} \in < \Phi >$$

Secondly we only need to prove that the number of elements in $\mathbb{Z}^n/\Phi$ and $\frac{2^{rn}}{det(\mathbb{B})}$ is the same. We prove it by induction.

– For $n = 2$, there is

$$\#\Phi = (2^r)^2, i.e. 2^{2r} \text{ n dimensional elements in } \mathbb{Z}^n,$$

since $det(B) = 1$, there is $\frac{2^{rn}}{det(\mathbf{B})} = 2^{2r}$ which is equal.

– For $n = 2^2$, there is

$$\#\Phi = (\frac{(2^{2r})}{|\phi|})^2 = \frac{2^{4r}}{2} = \frac{2^{rn}}{det(\mathbf{B})},$$

which is equal.

– For $n = 2^k$ where $k > 2$, Let $HW(\cdot)$ denote the Hamming weight, assume that the equality is true, we have

$$\#\Phi = \frac{(2^{rn/2})^2}{(|\phi|^{HW(0)+HW(1)+\cdots+HW(2^{k-1}-1)})^2} = \frac{2^{rn}}{det(\mathbf{B})},$$

then for $n = 2^{k+1}$, recall that $det(BW_n) = 2^{2^{(k-2)}}(det(BW_{n/2}))^2$, we have

$$\begin{aligned}
\frac{2^{r2^{k+1}}}{det(\mathbf{B})} &= \frac{2^{r2^k}}{det(BW_{n/2})} \cdot \frac{2^{r2^k}}{2^{2^{(k-2)}} det(BW_{n/2})} \\
&= \frac{2^{r2^k}}{(|\phi|^{HW(0)+HW(1)+\cdots+HW(2^{k-1}-1)})^2} \cdot \frac{2^{r2^k}}{2^{2^{k-2}}(|\phi|^{HW(0)+HW(1)+\cdots+HW(2^{k-1}-1)})^2} \\
&= \frac{2^{r2^k}}{(|\phi|^{HW(0)+HW(1)+\cdots+HW(2^{k-1}-1)})^2} \cdot \frac{2^{r2^k}}{(|\phi|^{2^{k-1}}|\phi|^{HW(0)+HW(1)+\cdots+HW(2^{k-1}-1)})^2} \\
&= \frac{2^{rn}}{(|\phi|^{HW(0)+HW(1)+\cdots+HW(2^{k-1}-1)+\cdots+HW(2^k-1)})^2}
\end{aligned} \tag{1}$$

Note that the last equation is true according to

$$HW(n/2 + i) = HW(n/2) + HW(i) \text{ for } 0 < i < n/2.$$

Therefore we finish our proof. □

Based on the Theorem 2, we give our whole encode process as shown in Algorithm 2.

Algorithm 2: Encoding Into BW Lattice Vector($\mathbf{m}$, $n = 2^k$)

Input: $m \in \{0,1\}^d$ where $d < log_2(\frac{2^{rn}}{det(B)})$
Output: $y \in BW_n \cap \mathbb{Z}^n_{2^r}$

$\mathbf{x} \in \mathbb{Z}^n/\Phi \longleftarrow m$
$y \longleftarrow$ Iterative Encoding($\mathbf{x}, n$)
return $\mathbf{y} \mod 2^r \in BW_n$

3.2 Decoding in the BW lattice code

In this subsection, we give our decode process. Given a vector in the $\mathbf{t} \in \mathbb{R}^n$, the BW decode problem is to find the closest lattice point to it.

$$\mathbf{t} \in \mathbb{R}^n \xrightarrow{\text{Step 0}} \mathbf{y} \in \mathbf{BW_n} \text{which is closest to t} \xrightarrow{\text{Step 1}} \mathbf{x} = \mathbf{B^{-1}y} \mod 2^r \xrightarrow{\text{Step 2}} \mathbf{m} \in \{0,1\}^d.$$

For the Step 0, we consider the decoding method of the bounded distance decoding problem shown in [24] . For the Step 1, we would apply the inverse iterative

Algorithm 3: Decoding in BW lattice($\mathbf{t} \in \mathbb{R}^n$, $n = 2^k$) [24]

Input: $\mathbf{t} = (\mathbf{t_1}, \mathbf{t_2}) \in \mathbb{R}^n$
Output: $\mathbf{y} \in BW_n$

$\mathbf{y}_1 \longleftarrow \text{BDD}(\mathbf{t_1}, BW_{n/2})$, $\mathbf{y}_2 \longleftarrow \text{BDD}(\mathbf{t_2}, BW_{n/2})$,
$\mathbf{y}_1^{(2)} \longleftarrow \text{BDD}(\mathbf{y}_1 - \mathbf{t_2}, \phi BW_{n/2})$, let $d_1 = dist((\mathbf{y}_1, y_1^{(2)} + \mathbf{t}_2), (\mathbf{t}_1, \mathbf{t}_2))$,
$\mathbf{y}_2^{(2)} \longleftarrow \text{BDD}(\mathbf{y}_2 - \mathbf{t_1}, \phi BW_{n/2})$, let $d_2 = dist((\mathbf{y}_2^{(2)} + \mathbf{t}_1, y_2), (\mathbf{t}_1, \mathbf{t}_2))$
if $d_1 < d_2$, **return** $(\mathbf{y}_1, y_1^{(2)} + \mathbf{t}_2)$
else, **return** $(\mathbf{y}_2^{(2)} + \mathbf{t}_1, y_2)$.

encoding process as shown in Algorithm 1 to get the $\mathbf{x}$, we give it in Algorithm 4.

Algorithm 4: Iterative Decoding($\mathbf{y}$, $n = 2^k$)

Input: $y \in BW_n$
Output: $x \in \mathbb{Z}^n$

$(x_1, x_2, \cdots, x_{n/2}) := (y_1 + iy_2, \cdots, y_{n/2-1} + iy_{n/2}) \hookleftarrow (y_1, y_2, \cdots, y_n)$
For $i = k-1, k-2, \cdots, 2, 1$
$(x_1, x_2, \cdots, x_{n/2}) \hookleftarrow (x_1, x_2, \cdots, x_{2^i}, [(x_{2^i+1}, x_{2^i+2}, \cdots, x_{2^{i+1}}) - (x_1, x_2, \cdots, x_{2^i})]/\phi \quad , \cdots, x_{2^k-2^i}, x_{2^k-2^i+1}, \cdots, x_{2^k-i}, [(x_{2^{k-i}+1}, x_{2^{k-i}+2}, \cdots, x_n) - (x_{2^{k-2i}}, x_{2^{k-2i}+1}, \cdots, x_{2^{k-i}})]/\phi)$
return $x \in \mathbb{Z}^n$

4 Algorithm Description

Then we combine the encoding method with the construction of a IND-CPA PKE scheme and finally give the IND-CCA KEM using the Fujisaki-Okamoto transformation.

4.1 Sub-Functions and Cryptographic Primitives

Scloud$^+$ make use of a pseudo-random function $\texttt{PRF} : \{0,1\}^{256} \to \{0,1\}^*$, two hash functions $\texttt{H} : \{0,1\}^* \to \{0,1\}^{256}$ and $\texttt{G} : \{0,1\}^* \to \{0,1\}^{256} \times \{0,1\}^{256}$, and a key-derivation function $\texttt{KDF} : \{0,1\}^* \to \{0,1\}^*$. The symmetric primitives $\texttt{PRF}, \texttt{H}, \texttt{G}$ and $\texttt{KDF}$ are instantiated as follows.

- $\texttt{H}$: SHA3-256;
- $\texttt{G}$: SHA3-512;
- $\texttt{KDF}$: SHAKE-256;
- $\texttt{PRF}(\mathbf{r})$: AES-256 in CTR mode, where the key is set to be $\mathbf{r}$, the nonce is set to be $\mathbf{0}$, and the counter is initialized to 0.

The sampling functions $\texttt{gen}, \phi, \psi$ and $\texttt{CenBinom}$ are specified as follows.

- $\texttt{gen}(\mathbf{seed_A})$ first generates a sequence of random integers $(t_0, t_1, \ldots, t_{mn-1}) \in \{0, 1, \ldots, q-1\}^*$ from the random coins $\mathbf{seed_A}$, and then returns a $m \times n$ matrix $\mathbf{A}$ which is filled by these integers.
- $\phi(\mathbf{r}, (m, n), h)$ first generates random integers $(t_0, t_1, \ldots) \in \{0, 1, \ldots, n-1\}^*$ from the random coins $\mathbf{r}$, and then determines a matrix $\mathbf{S}$ by Algorithm 5.
- $\psi(\mathbf{r}, (m, n), h)$ is computed similarly while interchanging the rows and columns.
- $\texttt{CenBinom}(\mathbf{r}, (m, n), \eta)$ first generates random bits $(t_0, t_1, \ldots, t_{2\eta mn-1}) \in \{0,1\}^*$, and then determines a matrix $\mathbf{E}$ by Algorithm 6.

4.2 Construction of Scloud$^+$.PKE

Scloud$^+$.PKE contains the following parameters.

- Modulus: powers of 2 integers q, q_k, q_1, q_2;
- Matrix size parameters: positive integers $m, n, \bar{m}, \bar{n}$;
- Secret weight parameters: h_s;
- Error parameter: η;
- Message length: $l_m \in \{128, 192, 256\}$;

Scloud$^+$.PKE includes three algorithms, i.e., the key generation (Algorithm 7), the encryption (Algorithm 8) and the decryption (Algorithm 9). The $\texttt{MsgEnc}$ and $\texttt{MsgDec}$ functions are defined based on specific parameters, which are detailed in Section 5.

Algorithm 5: The function $\phi(\mathbf{r}, (m, n), h)$

Input: A sequence of random integers $(t_0, t_1, \dots) \in \{0, 1, \dots, n-1\}^*$
Input: Positive integers m, n, h such that $n \geq 2h$
Output: An $m \times n$ matrix $\mathbf{S} \in H^{(m,n,h)}$

$\mathbf{S} = \mathbf{0}_{m\times n}$
$j = 0$
for i from 0 to $m-1$ **do**
  while $w_H(\mathbf{S}[i, \cdot]) < h$ **do**
    $\mathbf{S}[i, t_j] = -1$
    $j = j + 1$
  end while
  while $w_H(\mathbf{S}[i, \cdot]) < 2h$ **do**
    $\mathbf{S}[i, t_j] = 2 * \mathbf{S}[i, t_j] + 1$
    $j = j + 1$
  end while
end for
return $\mathbf{S}$

Algorithm 6: The function $\texttt{CenBinom}(\mathbf{r}, (m, n), \eta)$

Input: A sequence of random bits $(t_0, t_1, \dots, t_{2\eta mn-1}) \in \{0, 1\}^*$
Input: Positive integers m, n, η
Output: An $m \times n$ matrix $\mathbf{E}$

$\mathbf{E} = \mathbf{0}_{m\times n}$
$l = 0$
for i from 0 to $m-1$ **do**
  for j from 0 to $n-1$ **do**
    $\mathbf{E}[i][j] = \sum_{\alpha=0}^{\eta-1}(t_{l+2\alpha} - t_{l+2\alpha+1})$
    $l = l + 2\eta$
  end for
end for
return $\mathbf{E}$

Algorithm 7: Scloud$^+$.PKE.KeyGen()

Output: Public key $pk \in \mathbb{Z}_q^{m\times\bar{n}} \times \{0, 1\}^{256}$
Output: Secret key $sk \in \mathbb{Z}_q^{n\times\bar{n}}$

$\boldsymbol{\alpha} \hookleftarrow \{0, 1\}^{256}$
$(\mathbf{seed_A}, \mathbf{r}_1, \mathbf{r}_2) = \texttt{PRF}(\boldsymbol{\alpha}) \in \{0, 1\}^{256\times 3}$
$\mathbf{A} = \texttt{gen}(\mathbf{seed_A}) \in \mathbb{Z}_q^{m\times n}$
$\mathbf{S} = \psi(\mathbf{r}_1, (n, \bar{n}), h_s) \in \mathbb{Z}^{n\times\bar{n}}$, $\mathbf{E} = \texttt{CenBinom}(\mathbf{r}_2, (m, \bar{n}), \eta) \in \mathbb{Z}^{m\times\bar{n}}$
$\mathbf{B} = \mathbf{A} \cdot \mathbf{S} + \mathbf{E} \in \mathbb{Z}_q^{m\times\bar{n}}$
$\bar{\mathbf{B}} = \lfloor \frac{q_k}{q} \cdot \mathbf{B} \rceil$
return $pk = (\bar{\mathbf{B}}, \mathbf{seed_A})$, $sk = \mathbf{S}$

Algorithm 8: Scloud$^+$.PKE.Enc(pk, $\mathbf{m}$, $\mathbf{r}$)

Input: Public key $pk = (\bar{\mathbf{B}}, \mathbf{seed_A}) \in \mathbb{Z}_q^{m\times\bar{n}} \times \{0,1\}^{256}$
Input: Message $\mathbf{m} \in \{0,1\}^l$
Input: Random coins $\mathbf{r} \in \{0,1\}^{256}$
Output: Ciphertext $\mathbf{C} \in \mathbb{Z}_q^{\bar{m}\times(n+\bar{n})}$

$\mathbf{A} = \texttt{gen}(\mathbf{seed_A})$
$(\mathbf{r}_1, \mathbf{r}_2) = \texttt{PRF}(\mathbf{r}) \in \{0,1\}^{256\times 2}$
$\mathbf{S}' = \phi(\mathbf{r}_1, (\bar{m}, m), h_s) \in \mathbb{Z}^{\bar{m}\times m}$
$\mathbf{E}' = (\mathbf{E_1}, \mathbf{E_2}) = \texttt{CenBinom}(\mathbf{r}_2, (\bar{m}, n+\bar{n}), \eta)$, where $\mathbf{E}_1 \in \mathbb{Z}^{\bar{m}\times n}$, $\mathbf{E_2} \in \mathbb{Z}^{\bar{m}\times\bar{n}}$
$\boldsymbol{\mu} = \texttt{MsgEnc}(\mathbf{m}) \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$
$\mathbf{C}_1 = \mathbf{S}' \cdot \mathbf{A} + \mathbf{E}_1$, $\mathbf{C}_2 = \mathbf{S}' \cdot \bar{\mathbf{B}} + \mathbf{E}_2 + \boldsymbol{\mu}$
$\bar{\mathbf{C}}_1 = \lfloor \frac{q_1}{q} \cdot \mathbf{C}_1 \rceil$, $\bar{\mathbf{C}}_2 = \lfloor \frac{q_2}{q} \cdot \mathbf{C}_2 \rceil$
return $\mathbf{C} = (\bar{\mathbf{C}}_1, \bar{\mathbf{C}}_2)$

Algorithm 9: Scloud$^+$.PKE.Dec(sk, $\mathbf{C}$)

Input: Secret key $sk = \mathbf{S} \in \mathbb{Z}_q^{n\times\bar{n}}$
Input: Ciphertext $\mathbf{C} \in \mathbb{Z}_q^{\bar{m}\times(n+\bar{n})}$
Output: Message $\mathbf{m} \in \{0,1\}^l$

$\mathbf{C}'_1 = \lfloor \frac{q}{q_1} \cdot \bar{\mathbf{C}}_1 \rceil$, $\mathbf{C}'_2 = \lfloor \frac{q}{q_2} \cdot \bar{\mathbf{C}}_2 \rceil$
$\mathbf{D} = \mathbf{C}'_2 - \mathbf{C}'_1\mathbf{S} \in \mathbb{Z}_q^{\bar{m}\times\bar{n}}$
return $\mathbf{m} = \texttt{MsgDec}(\mathbf{D}) \in \{0,1\}^l$

Algorithm 10: Scloud$^+$.KEM.KeyGen()

Output: Public key $pk \in \mathbb{Z}_q^{m\times\bar{n}} \times \{0,1\}^{256}$
Output: Secret key $sk \in \mathbb{Z}_q^{n\times\bar{n}} \times \mathbb{Z}_q^{m\times\bar{n}} \times \{0,1\}^{256\times 3}$

$(pk, sk') = $ Scloud$^+$.PKE.KeyGen()
$\mathbf{hpk} = \texttt{H}(pk) \in \{0,1\}^{256}$
$\mathbf{z} \hookleftarrow \{0,1\}^{256}$
$sk = (sk', pk, \mathbf{hpk}, \mathbf{z})$
return (pk, sk)

Algorithm 11: Scloud$^+$.KEM.Encaps(pk)

Input: Public key $pk \in \mathbb{Z}_q^{m\times\bar{n}} \times \{0,1\}^{256}$
Output: Ciphertext $\mathbf{C} \in \mathbb{Z}_q^{\bar{m}\times(n+\bar{n})}$
Output: Shared session key $\mathbf{ss} \in \{0,1\}^*$

$\mathbf{m} \hookleftarrow \{0,1\}^l$
$(\mathbf{r}, \mathbf{k}) = \texttt{G}(\mathbf{m}||\texttt{H}(pk)) \in \{0,1\}^{256\times 2}$
$\mathbf{C} = $ Scloud$^+$.PKE.Enc$(pk, \mathbf{m}, \mathbf{r})$
$\mathbf{ss} = \texttt{KDF}(\mathbf{k}||\mathbf{C})$
return $(\mathbf{C}, \mathbf{ss})$

Algorithm 12: Scloud$^+$.KEM.Decaps()

Input: Ciphertext $\mathbf{C} \in \mathbb{Z}_q^{\bar{m}\times(n+\bar{n})}$
Input: Secret key $sk = (sk', pk, \mathbf{hpk}, \mathbf{z}) \in \mathbb{Z}_q^{n\times\bar{n}} \times \mathbb{Z}_q^{m\times\bar{n}} \times \{0,1\}^{256\times 3}$
Output: Shared session key $\mathbf{ss} \in \{0,1\}^*$

$\mathbf{m}' = \text{Scloud}^+.\text{PKE.Dec}(sk', \mathbf{C})$
$(\mathbf{r}', \mathbf{k}') = \mathtt{G}(\mathbf{m}'||\mathbf{hpk})$
$\mathbf{C}' = \text{Scloud}^+.\text{PKE.Enc}(pk, \mathbf{m}', \mathbf{r})$
if $\mathbf{C} = \mathbf{C}'$ **then**
  return $\mathbf{ss} = \mathtt{KDF}(\mathbf{k}, \mathbf{C})$
else
  return $\mathbf{ss} = \mathtt{KDF}(\mathbf{z}, \mathbf{C})$
end if

4.3 Construction of IND-CCA KEM

Scloud$^+$.KEM is obtained by apply the Fujisaki-Okamoto transformation to Scloud$^+$.PKE. Particularly, we follow the approach adopted in [3,25]. Scloud$^+$.KEM consists of three algorithms, i.e., key generation (Algorithm 10), encapsulation (Algorithm 11), and decapsulation (Algorithm 12).

5 Parameter Selection

We provide three parameter sets for Scloud$^+$, which are called Scloud$^+$-128, Scloud$^+$-192, and Scloud$^+$-256. The parameter sets are listed in table 1.

Table 1. Parameter sets of Scloud$^+$.

	l_m	(q, q_k, q_1, q_2)	(m, n)	$(\bar{m}, \bar{n})$	h_s	η
Scloud$^+$-128	128	$(4096, 512, 512, 256)$	$(640, 640)$	$(8, 8)$	160	1
Scloud$^+$-192	192	$(4096, 2048, 2048, 1024)$	$(900, 900)$	$(8, 8)$	225	1
Scloud$^+$-256	256	$(4096, 2048, 1024, 256)$	$(1120, 1120)$	$(12, 11)$	280	2

The `MsgEnc` and `MsgDec` Functions.

- Scloud$^+$-128: The 128-bit message is first divided into two 64-bit vectors, $\mathbf{m}_0$ and $\mathbf{m}_1$. Then, the iterative message encoding process described in Section 4 is applied to $\mathbf{m}_0$ and $\mathbf{m}_1$ to obtain two vectors, $\mathbf{v}_0$ and $\mathbf{v}_1$, in $\mathbb{Z}_q^{32}$. Finally, $\mathbf{v}_0$ and $\mathbf{v}_1$ are rearranged into an 8×8 matrix over $\mathbb{Z}_q$.
- Scloud$^+$-192: The 192-bit message is first divided into two 96-bit vectors, $\mathbf{m}_0$ and $\mathbf{m}_1$. Then, the iterative message encoding process described in Section 4 is applied to $\mathbf{m}_0$ and $\mathbf{m}_1$ to obtain two vectors, $\mathbf{v}_0$ and $\mathbf{v}_1$, in $\mathbb{Z}_q^{32}$. Finally, $\mathbf{v}_0$ and $\mathbf{v}_1$ are rearranged into an 8×8 matrix over $\mathbb{Z}_q$.

– Scloud$^+$-256: The 256-bit message is first divided into four 64-bit vectors, $\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$. Then, the iterative message encoding process described in Section 4 is applied to $\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ to obtain four vectors, $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$, in $\mathbb{Z}_q^{32}$. Finally, $\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3$ are rearranged into a 12×11 matrix over $\mathbb{Z}_q$.

References

1. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, Santa Fe, NM, USA, November 20–22, 1994. IEEE Computer Society Press.
2. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.
3. Michael Naehrig, Erdem Alkim, Joppe Bos, Léo Ducas, Karen Easterbrook, Brian LaMacchia, Patrick Longa, Ilya Mironov, Valeria Nikolaenko, Christopher Peikert, Ananth Raghunathan, and Douglas Stebila. FrodoKEM. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.
4. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.
5. Chris Peikert, Oded Regev, and Noah Stephens-Davidowitz. Pseudorandomness of ring-LWE for any ring and modulus. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 461–473, Montreal, QC, Canada, June 19–23, 2017. ACM Press.
6. Adeline Langlois and Damien Stehlé. Worst-case to average-case reductions for module lattices. *Des. Codes Cryptogr.*, 75(3):565–599, 2015.
7. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, and Damien Stehlé. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.
8. Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, Jose Maria Bermudo Mera, Michiel Van Beirendonck, and Andrea Basso. SABER. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.
9. Xianhui Lu, Yamin Liu, Dingding Jia, Haiyang Xue, Jingnan He, Zhenfei Zhang, Zhe Liu, Hao Yang, Bao Li, and Kunpeng Wang. LAC. Technical report, National Institute of Standards and Technology, 2019. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions.
10. Jiang Zhang, Yu Yu, Shuqin Fan, Zhenfeng Zhang, and Kang Yang. Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of

smaller sizes. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020: 23rd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 12111 of *Lecture Notes in Computer Science*, pages 37–65, Edinburgh, UK, May 4–7, 2020. Springer, Heidelberg, Germany.

11. Ronald Cramer, Léo Ducas, Chris Peikert, and Oded Regev. Recovering short generators of principal ideals in cyclotomic rings. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 559–585, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.
12. Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.
13. Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Short stickelberger class relations and application to ideal-SVP. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 324–348, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.
14. Léo Ducas, Maxime Plançon, and Benjamin Wesolowski. On the shortness of vectors to be found by the ideal-SVP quantum algorithm. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 322–351, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.
15. Ronald Cramer, Léo Ducas, and Benjamin Wesolowski. Mildly short vectors in cyclotomic ideal lattices in quantum polynomial time. *J. ACM*, 68(2):8:1–8:26, 2021.
16. Yanbin Pan, Jun Xu, Nick Wadleigh, and Qi Cheng. On the ideal shortest vector problem over random rational primes. In Anne Canteaut and François-Xavier Standaert, editors, *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 559–583. Springer, 2021.
17. Alice Pellet-Mary, Guillaume Hanrot, and Damien Stehlé. Approx-SVP in ideal lattices with pre-processing. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2019, Part II*, volume 11477 of *Lecture Notes in Computer Science*, pages 685–716, Darmstadt, Germany, May 19–23, 2019. Springer, Heidelberg, Germany.
18. Olivier Bernard and Adeline Roux-Langlois. Twisted-PHS: Using the product formula to solve approx-SVP in ideal lattices. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part II*, volume 12492 of *Lecture Notes in Computer Science*, pages 349–380, Daejeon, South Korea, December 7–11, 2020. Springer, Heidelberg, Germany.
19. Eric Stephen Barnes and Gordon Elliott Wall. Some extreme forms defined in terms of abelian groups. *Journal of the Australian Mathematical Society*, 1(1):47–63, 1959.
20. Elena Grigorescu and Chris Peikert. List decoding barnes-wall lattices. In *2012 IEEE 27th Conference on Computational Complexity*, pages 316–325. IEEE, 2012.
21. G. David Forney Jr. Coset codes-ii: Binary lattices and related codes. *IEEE Trans. Inf. Theory*, 34(5):1152–1187, 1988.
22. Vincent Corlay. *Decoding Algorithms for Lattices. (Algorithmes de décodage pour les réseaux de points)*. PhD thesis, Polytechnic Institute of Paris, France, 2020.

23. Daniele Micciancio and Antonio Nicolosi. Efficient bounded distance decoders for barnes-wall lattices. In Frank R. Kschischang and En-Hui Yang, editors, *2008 IEEE International Symposium on Information Theory, ISIT 2008, Toronto, ON, Canada, July 6-11, 2008*, pages 2484–2488. IEEE, 2008.
24. Vincent Corlay, Joseph J Boutros, Philippe Ciblat, and Loïc Brunel. On the decoding of barnes-wall lattices. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 519–524. IEEE, 2020.
25. Zhongxiang Zheng, Anyu Wang, Haining Fan, Chunhuan Zhao, Chao Liu, and Xue Zhang. Scloud: Public key encryption and key encapsulation mechanism based on learning with errors. *IACR Cryptol. ePrint Arch.*, page 95, 2020.