EMBLEM: (Ring) LWE-based Key Encapsulation With a New Multi-bit Encoding Method

Minhye Seo, Suhri Kim, Dong Hoon Lee, and Jong Hwan Park

***Abstract*—Lattice-based cryptography is a promising candidate for post-quantum cryptosystems, and a large amount of research has been conducted on learning with errors (LWE) problems, which are believed to be resistant against quantum attacks. In this paper, two new key encapsulation mechanisms (KEMs) are proposed, called EMBLEM and R.EMBLEM, based on (ring) LWE problems. The new KEMs have two main features: 1) Their security is based on the (ring) LWE problem with small secrets, which leads to both a secret key of constant size (regardless of the LWE parameters) and a relatively large standard deviation of the discrete Gaussian distributions. 2) They rely on a new multi-bit encoding method that is suitable for (ring) LWE-based encryption schemes. Compared to Regev's encoding method, the proposed method does not require any rounding operation for decoding, and in this sense, it is conceptually simpler and easier to understand. Concrete parameters of the KEMs targeting 128-bit security level (against classical attacks) are provided, and their performance is compared with that of previous (ring) LWE-based KEMs in the literature.**

***Index Terms*—lattice-based cryptography, chosen-ciphertext security, key encapsulation mechanism, small secret LWE**

I. Introduction

When quantum algorithms are put into practical use, currently used public key cryptosystems have proved to be no longer secure. Notably, Shor's algorithm [1] demonstrated that number-theoretic problems such as the integer factorization problem [2] and the discrete logarithm problem [3] can be solved in polynomial time using quantum algorithms. This had led to a full-scale study of quantum-resistant cryptosystems. Accordingly, the National Institute of Standard and Technology (NIST) [4] has begun the process of Post-Quantum Cryptography Standardization. Until now, there have been five main classes of post-quantum cryptography, namely, lattice-based, code-based, hash-based, multivariate-based, and isogeny-based. Lattice-based cryptography is considered one of the most promising branches, as worst-case to average-case reductions for (NP-hard) lattice problems are shown [5], [6] in a cryptographically important sense.

In 2005, Regev [6] introduced the learning with errors (LWE) problem as an average-case problem in lattice-based cryptography, and proposed the first public-key encryption scheme based on the LWE problem. Later, to make Regev's construction more efficient, Lindner and Peikert [7] suggested a public key encryption scheme where both a public key and a ciphertext are constructed as LWE instances. Since then, most of the LWE-based encryption schemes have followed this structural design. More precisely, the LP scheme is as follows: for some $m, n, q \in \mathbb{Z}^+$, a public key $(\mathbf{A}, \mathbf{B})$ consists of $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$ and $\mathbf{B} = \mathbf{A}\mathbf{x} + \mathbf{e} \in \mathbb{Z}_q^m$, and a secret key is $\mathbf{x}$, where each element of $\mathbf{x}$ and $\mathbf{e}$ is chosen from a discrete Gaussian distribution $\mathcal{GD}_s$ with a Gaussian parameter s[1]. That is, $\mathbf{x} \leftarrow \mathcal{GD}_s^n$ and $\mathbf{e} \leftarrow \mathcal{GD}_s^m$. A ciphertext $(\mathbf{c_0}, \mathbf{c_1})$ for a one-bit message $M \in \{0, 1\}$ consists of $\mathbf{c_0} = \mathbf{r^T A} + \mathbf{e_1} \in \mathbb{Z}_q^n$ and $\mathbf{c_1} = \mathbf{r^T B} + \mathbf{e_2} + \lfloor q/2 \rceil M \in \mathbb{Z}_q$. Here, the elements of all random vectors $\mathbf{r}$, $\mathbf{e_1}$, and $\mathbf{e_2}$ are chosen from the discrete Gaussian distribution $\mathcal{GD}_s$. In decryption, the message can be recovered by computing $\mathbf{c_1} - \mathbf{c_0}\mathbf{x} \in \mathbb{Z}_q$ and deciding whether the resulting value $(\mathbf{r^T e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}) + \lfloor q/2 \rceil M \in \mathbb{Z}_q$ is close to 0 (implying that $M = 0$) or to $\lfloor q/2 \rceil$ (implying that $M = 1$), i.e., by performing a rounding operation. Decryption requires the rounding operation because the error term $(\mathbf{r^T e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}) \in \mathbb{Z}_q$ does not disappear; moreover, for correct decryption, the absolute size of the error term should be smaller than $\lfloor q/4 \rceil$.

For practical use (under quantum computing), it is necessary to encrypt at least 256 bits for symmetric-key encryption, in which case the LP scheme can be used for multi-bit encryption by preparing $\mathbf{B}_i = \mathbf{A}\mathbf{x}_i + \mathbf{e}_i \in \mathbb{Z}_q^m$ and $\mathbf{x}_i \leftarrow \mathcal{GD}_s^n$ for $i = 1, \ldots, 256$ and running the scheme in parallel [8]. This naturally causes a size blowup in the public/secret key and ciphertext. Another approach is to encrypt a multi-bit (for instance, t-bit) message within a modulus q using the LP scheme, and if this is possible, the number of parallel executions of the basic LP scheme is reduced to $\lceil 256/t \rceil$. In the case of t-bit encryption, each t-bit message should be encoded by multiplying it by $\lfloor q/2^t \rceil$, and rounded to $\lfloor q/2^t \rceil$ during decryption. Then, the absolute size of the error term should be smaller than $\lfloor q/2^{t+1} \rceil$. However, the value t cannot be increased as desired. As t becomes larger, $\lfloor q/2^{t+1} \rceil$ becomes smaller, so that the size of the error term may exceed $\lfloor q/2^{t+1} \rceil$. That is, it may happen that $|\mathbf{r^T e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}| \geq \lfloor q/2^{t+1} \rceil$, and then the correctness of the LP scheme does not hold with high probability. Intuitively, in order for the multi-bit encryption scheme to work properly with a negligible correctness error (e.g., less than 2^{-128}), it is necessary to either increase the size of the modulus q or reduce the size of the error term. For the former case, it is not desirable to increase q excessively, because this results in a blowup of the public key size as well as slower operations in $\mathbb{Z}_q$. For the latter case, it is possible to choose $\mathbf{r} \leftarrow [-B, B]^m$ and $\mathbf{x} \leftarrow [-B, B]^n$ at random for a

M. Seo, S. Kim, and D.H. Lee are with the Graduate School of Information Security, Korea University, Seoul, Korea, e-mail: smh89122@korea.ac.kr, suhrikim@gmail.com, donghlee@korea.ac.kr

J.H. Park is with the Department of Computer Science, Sangmyung University, Seoul, Korea, e-mail: jhpark@smu.ac.kr

[1]The standard deviation ρ of $\mathcal{GD}_s$ is then $\rho = s/\sqrt{2\pi}$.

small positive integer B such as $B = 1$, instead of choosing them from $\mathcal{GD}_s^m$ and $\mathcal{GD}_s^n$. Such small secret vectors increase the dimension n (and thus m) by a factor of $O(\log(\log n))$ [9], [10] to provide the same security level as in $\mathcal{GD}_s$; nevertheless, the correctness error becomes smaller than the error that occurs in the case of $\mathcal{GD}_s$. FrodoKEM [11] is an example of the former case, where each element of $\mathbf{r}$ and $\mathbf{x}$ is still chosen from $\mathcal{GD}_s$ and a 4-bit message is encrypted at one LWE instance by increasing the size of modulus q.

In particular, choosing the secret vector $\mathbf{x} \leftarrow [-B, B]^n$ at random for a small B has two advantages. First, the secret key K can be of constant size, for instance 256 bits, regardless of the dimension n, because the vector $\mathbf{x}$ can be easily generated from K using a pseudo-random function (PRF). Secondly, the standard deviation $s/\sqrt{2\pi}$ of $\mathcal{GD}_s$ (that corresponds to the vectors $\mathbf{e}$, $\mathbf{e}_1$, and $\mathbf{e}_2$) can be increased to some extent (e.g., $s/\sqrt{2\pi} = 25$), whereas the correctness error remains negligible. This is because the error size is dominated by the two values $|\mathbf{r}^\mathbf{T}\mathbf{e}|$ and $|\mathbf{e_1}\mathbf{x}|$, and as $|\mathbf{r}|$ and $|\mathbf{x}|$ decrease, $|\mathbf{e}|$ and $|\mathbf{e_1}|$ can be increased. Such a large standard deviation ensures better security for an LWE-based encryption scheme than a small standard deviation. To further reduce the error size, several constructions use the sparse secret LWE [12]–[14], where the vectors $\mathbf{r}$ and $\mathbf{x}$ chosen from $[-1, 1]^m$ and $[-1, 1]^n$ have a predetermined small number h of nonzero elements and all the remaining elements are zero. Obviously, the sparse secret LWE is effective in reducing the size of the error term, but it provides considerably weaker security level than LWE with $\mathcal{GD}_s$ or $[-B, B]$. For example, based on the sparse secret LWE, HElib [15], [16] sets $n = 1024$ and $h = 64$ by default to provide 128-bit security, but [17] demonstrated that HElib offers approximately 73-bit security under the same parameter setting. In addition, [18] demonstrated that the cost of the primal attack (using BKZ) against the sparse secret LWE is considerably lower than against the small secret LWE.

A. Contributions

In this paper, we propose two new chosen-ciphertext-secure KEMs whose security relies on small secret (ring) LWE. The KEMs follow the same structural design as the LP construction, but there are two major differences: First, our basic construction is designed for encrypting multi-bit messages at a time by using small secret LWE. The elements of the vectors $\mathbf{r}$ and $\mathbf{x}$ in the proposed schemes are uniformly chosen from $[-B, B]$ at random for a small positive integer B, such as $B = 1$, and the size of the modulus q is moderately increased to encrypt t-bit message as desired. By using small secret LWE, the proposed KEMs naturally have two advantages, namely, the constant size of the secret key and the quite large standard deviation of $\mathcal{GD}_s$. Indeed, the size of the secret key is always 32 bytes in all parameter settings of the proposed KEMs and the standard deviation of $\mathcal{GD}_s$ can be set to $s/\sqrt{2\pi} = 25$ in the proposed LWE-based KEM. Moreover, by running the basic scheme in parallel, an encryption scheme can be constructed that handles at least 256-bit messages and is proved to be chosen-plaintext-secure under (ring) LWE. By applying the KEM variant of the Fujisaki–Okamoto (FO) transformation [19] to the chosen-plaintext-secure encryption schemes, chosen-ciphertext-secure KEMs under (ring) LWE can be finally constructed. With appropriate parameterization to at least 128-bit security level (against classical attacks), it will be shown that the proposed KEMs are favorably comparable to previous (ring) LWE-based KEMs [20]–[23] in terms of security and efficiency. In Section VII, the performance of the proposed KEMs are evaluated in each parameter set, and the results are compared with those of the previous (ring) LWE-based KEMs.

Secondly, the present construction uses a novel multi-bit encoding method that is suitable for (ring) LWE-based encryption. The encoding method (described below) is conceptually simpler and easier to understand for multi-bit encryption than the previous Regev's encoding method. The new encoding method is quite simple: 1) The t-bit message part and the d-bit error part are pre-divided within a modulus q for some (pre-defined) positive integers t and d, respectively, and 2) as shown in Fig. 1, the t-bit message is encoded by concatenating the bit strings $1||0^d$. In decryption, as long as the size of the error term is less than 2^d, that is, $|\mathbf{r}^\mathbf{T}\mathbf{e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}| < 2^d$, the error occurred does not affect the message part, and thus the t-bit message is extracted as it is. Therefore, the inequality is used for calculating the correctness error of decryption, and indeed using the equality, all KEMs are parameterized to provide approximately 2^{-140} probability of decryption failure. It should also be noticed that the message extraction does not require any rounding (or comparison) operation but simply reads the most significant t bits.

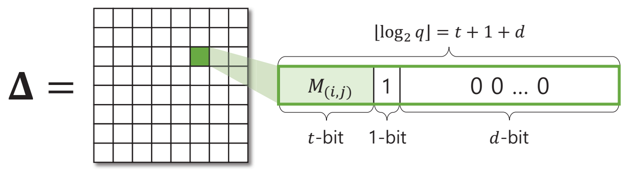


Fig. 1: Encoded message blocks

B. Overview of proposed encoding method

Fig. 1 shows the proposed encoding method for t-bit encryption within a modulus q. It should be noticed that for a given modulus q, an element in $\mathbb{Z}_q$ is represented as $\lfloor \log_2 q \rfloor + 1$ bits, and thus any $\lfloor \log_2 q \rfloor$-bit number belongs to $\mathbb{Z}_q$. The encoding is performed within $\lfloor \log_2 q \rfloor$ bits in $\mathbb{Z}_q$. It is assumed that a 256-bit message is represented by a $v \times k$ matrix $\mathbf{M} = \{M_{(i,j)}\}$ for some v, k such that $t \times v \times k = 256$, where each entry $M_{(i,j)}$ is t bits. By using the inequality $|\mathbf{r}^\mathbf{T}\mathbf{e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}| < 2^d$, the size d can be estimated in advance when the parameters are set, so as to ensure a negligible correctness error. Then, the length t of an encoded message is determined by the equation $\lfloor \log_2 q \rfloor = t + 1 + d$. Each t-bit message $M_{(i,j)}$ is encoded within the modulus q in the form of $M_{(i,j)}||1||0^d$, where the bit 1 in the middle is what is called "error-blocking bit". The resulting encoded messages are then reconstructed into a matrix $\mathbf{\Delta} \in \mathbb{Z}_q^{v \times k}$ again.

In the decryption phase, given the (matrix form of) ciphertext $(\mathbf{C_1}, \mathbf{C_2})$ and the secret key $\mathbf{X}$, $\mathbf{C_2} - \mathbf{C_1}\mathbf{X} =$

$\mathbf{E}$(Decryption error) + $\mathbf{\Delta}$(Encoded message) is computed, where each (i,j)-entry of $\mathbf{C_2} - \mathbf{C_1}\mathbf{X}$ is in the form $(\mathbf{r^T}\mathbf{e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}) + M_{(i,j)}||1||0^d$. If the error term ε is *positive* and less than 2^d, the error is added into the least significant d bits and has no effect on the message $M_{(i,j)}$, whereas if the error term ε is *negative* and its absolute value is less than 2^d, it affects beyond the least significant d bits. However, owing to the error-blocking bit 1, the negative error is computed as a positive value $2^d + \varepsilon$ $(< 2^d)$, and thus it is not propagated to the message part. Fig. 2 shows the cases of positive or negative error. As a result, any positive or negative error does not affect the message part as long as $|\mathbf{r^T}\mathbf{e} + \mathbf{e_2} - \mathbf{e_1}\mathbf{x}| < 2^d$.

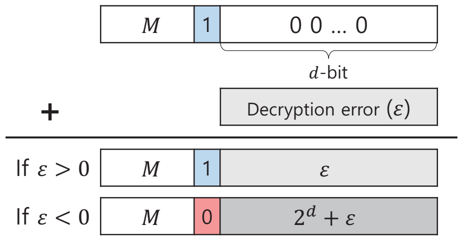


Fig. 2: Positive or negative error in decryption

C. Related work

Since Regev [6], [24] introduced the LWE problem and proved its worst-case to average-case (quantum) reduction, a large number of studies have been concerned with the construction of LWE-based cryptosystems. Peikert *et al.* [8] extended Regev's construction [6] to a multi-bit encryption scheme by simply running Regev's construction in parallel. However, as the security of Regev's encryption scheme was based on the LWE and the leftover hash lemma [25], it required a public key of fairly large size. To alleviate this problem, Lindner and Peikert [7] made the both public key and the ciphertext LWE-type instances, thereby eliminating the need of the leftover hash lemma. Accordingly, the security of the LP scheme depends solely on the LWE problem, which can reduce the size of the public key. Since then, a number of KEMs following the LP construction have been proposed [11], [26], [27].

In 2010, Lyubashevsky *et al.* [28] introduced the ring variant of the LWE problem, called *ring-LWE*. Despite the vulnerabilities inherent in the use of the ring structure [29]–[32], several cryptosystems have been proposed [23], [33]–[35] whose security relies on the hardness of ring-LWE because the ring structure allows to a significant efficiency improvement. Recently, an extended version of ring-LWE, called *module-LWE* [36], [37], has been developed, leading to several efficient KEMs [38]–[40]. A derandomized version of the LWE problem, called learning with rounding (LWR), was introduced in [41]. The LWR problem uses rounding to a smaller modulus $p(< q)$ [41]–[43] instead of adding Gaussian errors. Cheon *et al.* [44] proposed a KEM based on LWR, and D'Anvers *et al.* [13] suggested a KEM based on *module-LWR* in a ring setting. The advantage of LWR-based constructions is that Gaussian sampling of errors is no longer required in encryption, which allows faster encryption time, and the size of ciphertext transmitted can be smaller than in LWE-based schemes. However, the size of the decryption error in LWR-based schemes is considerably larger than in LWE-based schemes, even if the rounding modulus p is slightly smaller than the modulus q in terms of bit size (e.g,, $p = 2^{10}$ and $q = 2^{13}$). Thus, to reduce the decryption error, LWR-based schemes used the sparse secret LWR, which could cause security issues [17], [18], as mentioned above.

Compared to the present construction, all existing LWE- or LWR-based schemes have used a secret-key vector chosen from either the Gaussian or the sparse secret distributions, and none of them has ever used a secret-key vector chosen from the small secret distribution. Moreover, all the previous schemes have followed Regev's encoding method that requires the rounding operation. In this paper, we take advantage of both the small secret LWE and the novel (multi-bit) encoding method to construct the proposed KEMs, which are compared with previous (ring) LWE-based KEMs in terms of security and efficiency.

D. Organization

The remainder of this paper is organized as follows: In Section II, the basic notions are introduced, and the definitions of public-key encryption (PKE) and KEM, as well as their security models, are reviewed. In Section III, the new chosen-plaintext-secure PKE scheme and chosen-ciphertext-secure KEM based on the small secret LWE problem are presented. In Section IV, the ring variants of the constructions are presented. Section V is dedicated to analyzing the parameter selection. Finally, the subroutines of the implementation are described in Section VI, and the performance of the proposed schemes is analyzed in Section VII.

II. Background

A. Notation

The set of natural numbers is denoted by $\mathbb{N}$ and the set of integers by $\mathbb{Z}$. For $q \in \mathbb{N}$, the set $\mathbb{Z}_q$ is defined as $\mathbb{Z} \cap (-\frac{q}{2}, \frac{q}{2}]$. For a finite set $\mathcal{S}$, the notation $a \leftarrow_R \mathcal{S}$ implies that a is chosen uniformly at random from $\mathcal{S}$ and for a distribution $\mathcal{X}$, we write $a \leftarrow \mathcal{X}$ to denote that a is sampled according to the distribution $\mathcal{X}$. For a matrix $\mathbf{M}$, $\mathbf{M}^T$ denotes the transpose of $\mathbf{M}$. For a vector a of length n, a_i denotes the i-th component of a, and for a matrix $\mathbf{M}$, $\mathbf{M}[i,j]$ is the i-th row and j-th column entry of $\mathbf{M}$. The Euclidean norm $||a||$ is defined as $\sqrt{\Sigma_{i=1}^n a_i^2}$. For a bit string b, a function $[b]_t$ is defined that outputs the most significant t bits of b. This function is extended to matrices by applying it to each component of a matrix. For positive integers q and n, $\mathcal{U}(\mathbb{Z}_q^n)$ is defined by the uniform distribution over $\mathbb{Z}_q^n$. Throughout this paper, $\log$ is the logarithm with base 2.

B. Discrete Gaussian Distribution

For a given $s > 0$, the *discrete* Gaussian distribution over a lattice L is defined as

$$\mathcal{GD}_{L,s}(x) = \frac{\rho_s(x)}{\Sigma_{y \in L}\rho_s(y)} \qquad (1)$$

for any $x \in L$, where ρ denote the Gaussian function $\rho_s(x) = e^{-\pi\|x\|^2/s^2}$.

Note that the standard deviation of $\mathcal{GD}_{L,s}$ is $\sigma = s/\sqrt{2\pi}$. The Gaussian parameter s is used to describe a discrete Gaussian distribution. $\mathcal{GD}_s$ denotes the discrete Gaussian distribution $\mathcal{GD}_{\mathbb{Z},s}$. Moreover, it holds that $\mathcal{GD}_{\mathbb{Z}^n,s} = \mathcal{GD}_s^n$.

C. Learning with Errors

The learning with errors (LWE) problem with multiple secrets will now be defined, which has been proven to be at least as hard as the standard LWE problem [8].

Definition 2.1 (Decision LWE problem). Let $m, n, k, q \in \mathbb{N}$ and $\mathcal{D}_s, \mathcal{D}_e$ be distributions over $\mathbb{Z}_q$. One is given an instance $(\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times k}$ and is asked to distinguish whether there exists $\mathbf{S} \leftarrow \mathcal{D}_s^{n\times k}$ such that the instance is of the form $(\mathbf{A}, \mathbf{AS} + \mathbf{E} \mod q)$ with $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{m\times n}$, $\mathbf{E} \leftarrow \mathcal{D}_e^{m\times k}$ or the instance is uniformly chosen at random from $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times k}$. The decision LWE problem is denoted by $\mathsf{LWE}_{n,m,q,\mathcal{D}_e}$.

The binary-LWE problem (where the secret matrix $\mathbf{S}$ is randomly chosen from $\{-1,0,1\}^{n\times k}$) has been considered in [45], [46]. In [9], Bai and Galbraith proved that the binary-LWE problem is as hard as the standard LWE problem as long as the parameter n is increased by a factor of $\log(\log n)$. Note that the errors are still chosen from discrete Gaussian distributions. Albrecht *et al.* [10] applied the embedding technique of Bai and Galbraith [9] to the generalized cases, where the elements of $\mathbf{S}$ are randomly sampled from a small interval. Here, [10] is applied to the case where $\mathbf{S}$ is sampled from $[-B, B]^{n\times k}$ for any positive integer $B < \sigma$. The decisional small secret LWE problem, denoted by $\mathsf{smaLWE}_{n,m,q,\mathcal{D}_e}$, is defined as follows.

Definition 2.2 (Decision LWE Problem with small secrets [45]). Let $m, n, k, q \in \mathbb{N}$ and $\mathcal{D}_e$ be a distribution over $\mathbb{Z}_q$. One is given an instance $(\mathbf{A}, \mathbf{B}) \in \mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times k}$ and is asked to distinguish whether there exists $\mathbf{S} \leftarrow_R [-B, B]^{n\times k}$ such that the instance is of the form $(\mathbf{A}, \mathbf{AS}+\mathbf{E} \mod q)$ with $\mathbf{A} \leftarrow_R \mathbb{Z}_q^{m\times n}$, $\mathbf{E} \leftarrow \mathcal{D}_e^{m\times k}$ or the instance is uniformly chosen at random from $\mathbb{Z}_q^{m\times n} \times \mathbb{Z}_q^{m\times k}$.

Lemma 2.3 (Log root Hermite factor of LWE instances of small secret [10]). Let a small secret LWE instance be characterized by n, α, q, let $\mathbf{s}_{(i)} \leftarrow_R [-B, B]$, let $\xi = 1/B$, and let $\sigma = \frac{\alpha q}{\sqrt{2\pi}}$. Any lattice reduction algorithm achieving *log root-Hermite factor*

$$\log\delta = \frac{\big(\log(q/\sigma) - \log(2\tau\sqrt{\pi e})\big)^2 \cdot \log(q/\sigma)}{n\big(2\log(q/\sigma) - \log\xi\big)^2} \qquad (2)$$

solves LWE by reducing BDD to uSVP for some fixed $\tau \le 1$ if $(q^m(\xi\sigma)^n)^{1/(m+n)} \cdot \sqrt{\frac{m+n}{2\pi e}} \le q$, where $m = m' - n = \sqrt{\frac{n(\log q - \log\sigma)}{\log\delta}} - n$.

Lyubashevsky *et al.* [28] proposed the ring-LWE problem over rings. The notation $a \leftarrow \mathcal{X}^n$ implies that n coefficients of a polynomial a are chosen independently from $\mathcal{X}$.

Definition 2.4 (Decision Ring-LWE problem). Let $n, q \in \mathbb{N}$ and $\mathcal{D}_s, \mathcal{D}_e$ be distributions over $\mathbb{Z}_q$. For an irreducible polynomial $f(x) \in \mathbb{Z}[x]$ of degree n, let $R_q = \mathbb{Z}_q[x]/f(x)$ be the ring modulo q. One is given $(a, b) \in R_q^2$ and is asked to distinguish whether there exists $s \leftarrow \mathcal{D}_s^n$ such that (a, b) is of the form $(a, a \cdot s + e)$ with $a \leftarrow_R R_q$, $e \leftarrow \mathcal{D}_e^n$ or (a, b) is chosen uniformly at random from R_q^2. The decision Ring-LWE problem is denoted by $\mathsf{RLWE}_{n,q,\mathcal{D}_e}$.

The Ring-LWE problem with small secrets (where the coefficients of the secret polynomial s are randomly chosen from $[-B, B]^n$) can be defined as in the case of smaLWE. In [47], it is shown that for the standard LWE, the secret s can be sampled from any distribution as long as its entropy is sufficiently large. Under the assumption that the analysis in the standard LWE setting equally holds in the ring-LWE setting, the coefficients of the secret polynomial can be set to be small in the ring-LWE problem. The decisional small secret Ring-LWE problem, denoted by $\mathsf{smaRLWE}_{n,q,\mathcal{D}_e}$, is defined as follows.

Definition 2.5 (Decision Ring-LWE Problem with small secrets). Let $n, q \in \mathbb{N}$ and $\mathcal{D}_e$ be a distribution over $\mathbb{Z}_q$. For an irreducible polynomial $f(x) \in \mathbb{Z}[x]$ of degree n, let $R_q = \mathbb{Z}_q[x]/f(x)$ be the ring modulo q. One is given $(a, b) \in R_q^2$ and is asked to distinguish whether there exists $s \leftarrow_R [-B, B]^n$ such that (a, b) is of the form $(a, a \cdot s + e)$ with $a \leftarrow_R R_q$, $e \leftarrow \mathcal{D}_e^n$ or (a, b) is uniformly chosen at random from R_q^2.

D. Definitions

Definition 2.6 (Public Key Encryption). A public-key encryption (PKE) scheme consists of the following three algorithms: KeyGen, $\mathsf{Encrypt}$, and $\mathsf{Decrypt}$.

- $\mathsf{KeyGen}(1^\lambda)$: The key generation algorithm takes as input the security parameter 1^λ and outputs a public/secret key pair (pk, sk).
- $\mathsf{Encrypt}(pk, M)$: The encryption algorithm takes as input a public key pk and a message $M \in \mathcal{M}$. Then it outputs a corresponding ciphertext C.
- $\mathsf{Decrypt}(sk, C)$: The decryption algorithm takes as input a secret key sk and a ciphertext C. It outputs a message M or $\perp$ (which indicates decryption failure).

Correctness. The correctness of a PKE scheme is ensured if the following condition holds: For all $M \in \mathcal{M}$,

$$\Pr\big[(pk, sk) \leftarrow_R \mathsf{KeyGen}(1^\lambda);\ C \leftarrow_R \mathsf{Encrypt}(pk, M) : \mathsf{Decrypt}(sk, C) = M\big] > 1 - \epsilon(\lambda),$$

where ϵ is a negligible function.

Definition 2.7 (Key Encapsulation Mechanism). A key encapsulation mechanism (KEM) consists of the following three algorithms: KeyGen, Encap, and Decap.

- $\mathsf{KeyGen}(1^\lambda)$: The key generation algorithm takes as input the security parameter 1^λ and outputs a public/secret key pair (pk, sk).

- $\mathsf{Encap}(pk)$: The encapsulation algorithm takes as input a public key pk. Then it outputs a ciphertext C and a key $K \in \mathcal{K}$.
- $\mathsf{Decap}(sk, C)$: The decapsulation algorithm takes as input a secret key sk and a ciphertext C. It outputs a key K or $\perp$ (which indicates decapsulation failure).

Correctness. The correctness of KEM is ensured if the following condition holds:

$$\Pr\left[(pk, sk) \leftarrow_R \mathsf{KeyGen}(1^\lambda);\ (C, K) \leftarrow_R \mathsf{Encap}(pk) : \mathsf{Decap}(sk, C) = K\right] > 1 - \epsilon(\lambda),$$

where ϵ is a negligible function.

Definition 2.8 (IND-CPA Security of PKE). Let PKE = (KeyGen, Encrypt, Decrypt) be a public-key encryption scheme. An experiment (parameterized by a bit b) between an adversary $\mathcal{A}$ and a challenger is defined as follows:

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

1. *The challenger runs* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *outputs two messages* (M_0, M_1) *of the same length.*
3. *The challenger computes* $\mathsf{Encrypt}(pk, M_b)$ *for a randomly chosen bit* $b \in \{0, 1\}$ *and gives it to* $\mathcal{A}$.
4. $\mathcal{A}$ *outputs a bit* b'. *The challenger returns* b' *as the output of the game.*

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

$$\mathbf{Adv}^{\text{IND-CPA}}_{\mathsf{PKE},\mathcal{A}} = \Big| \Pr\left[\textbf{IND-CPA}^1_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\textbf{IND-CPA}^0_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1\right] \Big|.$$

We say that PKE *is IND-CPA secure if for any polynomial time adversary* $\mathcal{A}$ *and any* λ, *we have* $\mathbf{Adv}^{\textit{IND-CPA}}_{\mathsf{PKE},\mathcal{A}} \le \epsilon(\lambda)$, *where* ϵ *is a negligible function.*

Definition 2.9 (IND-CCA Security of KEM). Let KEM = (KeyGen, Encap, Decap) be a key encapsulation mechanism. An experiment (parameterized by a bit b) between an adversary $\mathcal{A}$ and a challenger is defined as follows:

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

1. *The challenger runs* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$.
2. $\mathcal{A}$ *queries the decapsulation oracle* $\mathsf{Decap}(sk, \cdot)$.
3. *The challenger computes* $(C^*, K_0^*) \leftarrow \mathsf{Encap}(pk)$ *and* $K_1^* \leftarrow_R \mathcal{K}$. *Then the challenger gives* (C^*, K_b^*) *to* $\mathcal{A}$ for a randomly chosen bit $b \in \{0, 1\}$.
4. $\mathcal{A}$ *continues to query the decapsulation oracle but may not query the ciphertext* C^*. *Finally,* $\mathcal{A}$ *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

$$\mathbf{Adv}^{\text{IND-CCA}}_{\mathsf{KEM},\mathcal{A}} = \Big| \Pr\left[\textbf{IND-CCA}^1_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\textbf{IND-CCA}^0_{\mathsf{KEM},\mathcal{A}}(\lambda) = 1\right] \Big|.$$

We say that KEM *is IND-CCA secure if for any polynomial time adversary* $\mathcal{A}$ *and any* λ, *we have* $\mathbf{Adv}^{\textit{IND-CCA}}_{\mathsf{KEM},\mathcal{A}} \le \epsilon(\lambda)$, *where* ϵ *is a negligible function.*

In the (quantum) random oracle model, the adversary additionally issues the (quantum) random oracles. The constructions in this paper will consider the quantum accessible random oracles described in [48].

III. CCA-Secure Key Encapsulation Mechanism

In this section, we introduce an IND-CCA secure KEM, called EMBLEM. An IND-CPA secure PKE scheme, EMBLEM.CPA, is first constructed, and then it is converted into EMBLEM using the KEM variant [19] of the Fujisaki–Okamoto transformation.

A. EMBLEM.CPA (CPA-Secure PKE)

• **Encoding and decoding functions.** Let $\mathcal{M} = \{0, 1\}^\ell$ be the message space. The encoding function **`encode`** is first defined; it takes an ℓ-bit string as input and outputs the encoded message in matrix form. The decoding function **`decode`** is an inverse of **`encode`**. For an ℓ-bit message M, a block size t, an upper bound d for the error size, positive integers v and k such that $\ell/t = v \times k$, and a modulus q, **`encode`** and **`decode`** are defined as follows:

▶ **`encode`**(M, t, d, v, k, q)

1) Split the message into t-bit blocks (it is assumed that t divides ℓ) and generate message blocks $\{M_{i'}\}$ for $i' \in [1, \ell/t]$.
2) Compute $M_{(i,j)} = M_{i'}$ for $i' \in [1, \ell/t]$, where $i = \lceil i'/k \rceil \in [1, v]$ and $j = i' - \lfloor i'/k \rfloor \cdot k \in [1, k]$.
3) Output a $v \times k$ matrix $\mathbf{M} = \{\mathbf{M}[i, j]\}$, where $\mathbf{M}[i, j] \leftarrow M_{(i,j)}||1||0^d$ for $i \in [1, v]$, $j \in [1, k]$ (as in Fig. 3).

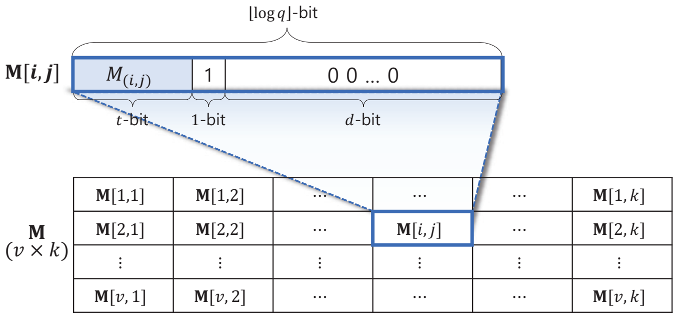


Fig. 3: Encoding function **`encode`**

▶ **`decode`**$(\mathbf{M}, t, q)$

1) Compute $M_{i'} = \left[\mathbf{M}[i, j]\right]_t$ for $i \in [1, v], j \in [1, k]$, where $i' = (i - 1)k + j \in [1, \ell/t]$;
2) Output ℓ-bit string $M = M_1|| \cdots ||M_{\ell/t}$.

• **Sampling function.** The sampling function **`Sam`** is now defined; it takes as input a random seed r and outputs ephemeral values $(\mathbf{R}, \mathbf{E_1}, \mathbf{E_2})$, where $\mathbf{R}$ is a random matrix in $[-B, B]^{m \times v}$ and $(\mathbf{E_1}, \mathbf{E_2})$ is randomly sampled from $\mathcal{GD}_s^{v \times (n+k)}$ for some s. The function **`Sam`** produces a deterministic output for the same input, and the size of the outputs $(\mathbf{R}, \mathbf{E_1}, \mathbf{E_2})$ can be variable depending on a security parameter. In practice, a Gaussian sampler internally selects a random seed and uses it to generate a sampling output. In the **Sam** function, s_1 and s_2 are internally computed as $s_1 \leftarrow \text{PRF}(r||1)$ and $s_2 \leftarrow \text{PRF}(r||2)$ and are used for generating $\mathbf{E_1}$ and $\mathbf{E_2}$, respectively.

• **Scheme.** Given a security parameter 1^λ, the system-wide parameter $params$ is generated as follows: Positive integers m, n, k, t, d, v and the modulus q are chosen under the constraint that $m > n$ and $m > n+k$. A standard deviation $\sigma = s/\sqrt{2\pi}$ for $\mathcal{GD}_s$ and a positive integer $B < \sigma$ are chosen. Then, $params$ is given by $(m, n, k, q, t, d, v, B, \mathcal{GD}_s)$. Note that $\ell/t = v \times k$. It is assumed that $params$ is used for all algorithms in EMBLEM.CPA.

KeyGen(1^λ): Choose a random matrix $\mathbf{A} \leftarrow \mathbb{Z}_q^{m\times n}$, a secret random matrix $\mathbf{X} \leftarrow [-B, B]^{n\times k}$, and an error matrix $\mathbf{E} \leftarrow \mathcal{GD}_s^{m\times k}$. Compute $\mathbf{B} = \mathbf{AX} + \mathbf{E}$. The key pair (pk, sk) is given by $pk = (\mathbf{A}, \mathbf{B})$ and $sk = (\mathbf{X})$. [2]

Encrypt(pk, M): To encrypt a message, proceed as follows:

1) $\mathbf{M} \leftarrow \mathbf{encode}(M, t, d, v, k, q)$.
2) Choose a random seed $r \in \{0, 1\}^{256}$.
3) $(\mathbf{R}, \mathbf{E_1}, \mathbf{E_2}) \leftarrow \mathbf{Sam}(r)$, where $\mathbf{R} \in [-B, B]^{m\times v}$ and $(\mathbf{E_1}, \mathbf{E_2}) \in \mathcal{GD}_s^{v\times(n+k)}$.
4) Compute $(\mathbf{C_1}, \mathbf{C_2}) = (\mathbf{R}^T\mathbf{A} + \mathbf{E_1}, \mathbf{R}^T\mathbf{B} + \mathbf{E_2} + \mathbf{M})$.
5) Return the ciphertext $C = (\mathbf{C_1}, \mathbf{C_2}) \in \mathbb{Z}_q^{v\times n} \times \mathbb{Z}_q^{v\times k}$.

Decrypt(sk, C): Parse the ciphertext C as $(\mathbf{C_1}, \mathbf{C_2})$.

1) Compute $\mathbf{M} = \mathbf{C_2} - \mathbf{C_1}\mathbf{X}$.
2) Output $M \leftarrow \mathbf{decode}(\mathbf{M}, t, q)$.

Theorem 3.1 (Correctness). *For simplicity, the case $v = 1$ is considered. Then $\mathbf{R} \in [-B, B]^m$, $\mathbf{E_1} \in \mathcal{GD}_s^n$, and $\mathbf{E_2} \in \mathcal{GD}_s^k$. For $i \in [1, k]$, the following are defined:*

- $\mathbf{E} = \{E^{(i)}\}$, where $E^{(i)} \in \mathcal{GD}_s^m$ is an i-th column of $\mathbf{E}$.
- $\mathbf{E_2} = \{e_2^{(i)}\}$, where $e_2^{(i)} \in \mathcal{GD}_s$ is an i-th entry of $\mathbf{E_2}$.
- $\mathbf{X} = \{X^{(i)}\}$, where $X^{(i)} \in [-B, B]^n$ is an i-th column of $\mathbf{X}$.

Let $\epsilon = \Pr\left[\max_{i\in[1,k]}\left\{|\langle\mathbf{R}, E^{(i)}\rangle| + |e_2^{(i)}| + |\langle\mathbf{E_1}, X^{(i)}\rangle|\right\} \geq 2^d\right]$, *where $d = \lfloor \log q \rfloor - (t+1)$. Then* EMBLEM.CPA *is $(1-\epsilon)$-correct.*

Proof. The decryption proceeds as follows:

$$\begin{aligned}\mathbf{C_2} - \mathbf{C_1}\mathbf{X} &= (\mathbf{R}^T\mathbf{B} + \mathbf{E_2} + \mathbf{M}) - (\mathbf{R}^T\mathbf{A} + \mathbf{E_1})\mathbf{X} \\ &= \big(\mathbf{R}^T(\mathbf{AX} + \mathbf{E}) + \mathbf{E_2} + \mathbf{M}\big) - (\mathbf{R}^T\mathbf{A} + \mathbf{E_1})\mathbf{X} \\ &= (\mathbf{R}^T\mathbf{E} + \mathbf{E_2} - \mathbf{E_1}\mathbf{X}) + \mathbf{M}.\end{aligned}$$

Note that for $j \in [1, k]$, the j-th entry of $(\mathbf{R^T E} + \mathbf{E_2} - \mathbf{E_1 X}) \in \mathbb{Z}_q^k$ becomes $\langle\mathbf{R}, E^{(j)}\rangle + e_2^{(i)} - \langle\mathbf{E_1}, X^{(j)}\rangle$, and the j-th entry of $\mathbf{M} \in \mathbb{Z}_q^k$ is in the form $M_j||1||0^d$, where $M_j \in \{0, 1\}^t$ is the j-th message block of M. Therefore, if the absolute size of the j-th entry in $(\mathbf{R^T E} + \mathbf{E_2} - \mathbf{E_1 X})$ is less than 2^d, it does not affect the message block M_j (mentioned in Section I-B). As a result, the scheme satisfies correctness with probability of $1 - \epsilon$. □

In Section V, the parameters will be set so that the decryption error ϵ may be negligible, i.e., $\epsilon = 2^{-140}$.

Theorem 3.2 (Security). *EMBLEM.CPA is IND-CPA secure if the smaLWE$_{n,m,q}$ assumption holds.*

Proof. The proof proceeds by a sequence of games. In Game 0, the public key is a small secret LWE instance and the ciphertext is an encryption of M_0. In Game 5, the public key is a small secret LWE instance and the ciphertext is an encryption of M_1. It will be shown that the distributions between Game 0 and Game 5 are computationally indistinguishable for the adversary.

▷ **Game 0.** This is the real game, where the public key and the ciphertext are generated honestly as in the real construction. In this game, the ciphertext is an encryption of M_0. Note that $\mathbf{M_0} \leftarrow \mathbf{encode}(M_0, t, d, v, k, q)$. The distribution of Game 0, denoted by $\mathcal{D}_0$, is given as follows:

$$\begin{aligned}\mathcal{D}_0 = \{&pk \leftarrow (\mathbf{A}, \mathbf{B} = \mathbf{AX} + \mathbf{E}),\\ &C \leftarrow (\mathbf{C_1} = \mathbf{R^T A} + \mathbf{E_1}, \mathbf{C_2} = \mathbf{R^T B} + \mathbf{E_2} + \mathbf{M_0})\}.\end{aligned}$$

▷ **Game 1.** In Game 1, the public key $\mathbf{B}$ is uniformly generated at random. The remaining part is the same as in Game 0. The distribution of Game 1, denoted by $\mathcal{D}_1$, is given as follows:

$$\begin{aligned}\mathcal{D}_1 = \{&pk \leftarrow (\mathbf{A}, \boxed{\mathbf{B} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m\times k})}),\\ &C \leftarrow (\mathbf{C_1} = \mathbf{R^T A} + \mathbf{E_1}, \mathbf{C_2} = \mathbf{R^T B} + \mathbf{E_2} + \mathbf{M_0})\}.\end{aligned}$$

▷ **Game 2.** In Game 2, small secret LWE instances in the ciphertext are changed into random matrices. That is, the ciphertext $C = (\mathbf{C_1}, \mathbf{C_2}) = (\mathbf{U_1}, \mathbf{U_2} + \mathbf{M_0})$, where $(\mathbf{U_1}, \mathbf{U_2})$ is uniformly generated at random in $\mathbb{Z}_q^{v\times n} \times \mathbb{Z}_q^{v\times k}$. The remaining part is the same as in Game 1. The distribution of Game 2, denoted by $\mathcal{D}_2$, is given as follows:

$$\begin{aligned}\mathcal{D}_2 = \{&pk \leftarrow (\mathbf{A}, \mathbf{B} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m\times k})),\\ &C \leftarrow (\boxed{\mathbf{C_1} = \mathbf{U_1}, \mathbf{C_2} = \mathbf{U_2} + \mathbf{M_0}})\}.\end{aligned}$$

▷ **Game 3.** In Game 3, the message M_0 changes to another message M_1. Note that $\mathbf{M_1} \leftarrow \mathbf{encode}(M_1, t, d, v, k, q)$. The remaining part is the same as in Game 2. The distribution of Game 3, denoted by $\mathcal{D}_3$, is given as follows:

$$\begin{aligned}\mathcal{D}_3 = \{&pk \leftarrow (\mathbf{A}, \mathbf{B} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m\times k})),\\ &C \leftarrow (\boxed{\mathbf{C_1} = \mathbf{U_1}, \mathbf{C_2} = \mathbf{U_2} + \mathbf{M_1}})\}.\end{aligned}$$

▷ **Game 4.** In Game 4, the ciphertext is restored to small secret LWE instances, and the remaining part is the same as in Game 3. The distribution of Game 4, denoted by $\mathcal{D}_4$, is given as follows:

$$\begin{aligned}\mathcal{D}_4 = \{&pk \leftarrow (\mathbf{A}, \mathbf{B} \leftarrow \mathcal{U}(\mathbb{Z}_q^{m\times k})),\\ &C \leftarrow (\boxed{\mathbf{C_1} = \mathbf{R^T A} + \mathbf{E_1}, \mathbf{C_2} = \mathbf{R^T B} + \mathbf{E_2} + \boxed{\mathbf{M_1}}})\}.\end{aligned}$$

▷ **Game 5.** In Game 5, the public key is restored to the small secret LWE instance, and the remaining part is the same as in

[2]The secret matrix $\mathbf{X}$ can be generated using a PRF and a short key $seed_{\mathbf{X}}$, so that a user need only store $seed_{\mathbf{X}}$ instead of the entire matrix $\mathbf{X}$. Similarly, the matrix $\mathbf{A}$ in the public key can also be derived from a seed by using PRF. In [11], `AES128-ECB` was used as a PRF with a 256-bit seed.

Game 4. The distribution of Game 5, denoted by $\mathcal{D}_5$, is given as follows:

$$\mathcal{D}_5 = \{pk \leftarrow (\mathbf{A}, \boxed{\mathbf{B} = \mathbf{AX} + \mathbf{E}}), \\ C \leftarrow (\mathbf{C_1} = \mathbf{R^T A} + \mathbf{E_1}, \mathbf{C_2} = \mathbf{R^T B} + \mathbf{E_2} + \mathbf{M_1})\}.$$

The distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable under the $\mathsf{smaLWE}_{n,m,q}$ assumption. The distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ are also computationally indistinguishable under the $\mathsf{smaLWE}_{m,n+k,q}$ assumption, as the ciphertext in Game 1 forms the small secret LWE instances with $(n+k)$ samples in dimension m. Since $m > n$ and $m > n+k$ in the $params$ generation, the $\mathsf{smaLWE}_{n,m,q}$ is reduced to the $\mathsf{smaLWE}_{m,n+k,q}$, and thus only the $\mathsf{smaLWE}_{n,m,q}$ is sufficient for the security assumption. In Games 2 and 3, the ciphertexts are computed in a one-time pad manner by adding the message to a random matrix, so that the distributions $\mathcal{D}_2$ and $\mathcal{D}_3$ are statistically indistinguishable. The hybrid games, i.e., Games 3–5, proceed as in the case of Games 0–2, but in a reverse manner. If the parameters n, m and q of the smaLWE are set to be as hard as those of LWE, then the security of the standard LWE problem can be reduced to that of EMBLEM.CPA. □

B. *EMBLEM (CCA-Secure KEM)*

In this section, we propose an IND-CCA secure KEM, called EMBLEM, in the quantum random oracle model. To achieve CCA security, the KEM variant on the FO transformation [19] is applied to the present IND-CPA secure encryption scheme EMBLEM.CPA.

Let $\mathcal{M} = \{0,1\}^{256}$ be the message space of EMBLEM. CPA. The system parameter $params$ is given as in EMBLEM.CPA. In the FO transformation, the following three hash functions are required:

- The hash function $G : \{0,1\}^* \rightarrow \{0,1\}^{256}$.
- The hash function $H : \{0,1\}^* \rightarrow \{0,1\}^{256}$.
- The hash function $\hat{H} : \{0,1\}^* \rightarrow \{0,1\}^{256}$.

These hash functions will be modeled as random oracles in the security proof. The CCA-secure KEM EMBLEM = (KeyGen, Encap, Decap) is constructed as follows:

KeyGen(1^λ): Same as EMBLEM.CPA.KeyGen.

Encap(pk): To generate a key K and a ciphertext C, proceed as follows:

1) Select a random string $\delta \leftarrow \{0,1\}^{256}$ and compute $r = G(\delta)$.
2) Compute $C_1 \leftarrow$ EMBLEM.CPA.Encrypt($pk, \delta; r$) and $C_2 = \hat{H}(\delta)$.
3) Compute $K = H(\delta, C_1, C_2)$.
4) Return the ciphertext $C = (C_1, C_2) \in \mathbb{Z}_q^{v\times(n+k)} \times\{0,1\}^{256}$ and the key $K \in \{0,1\}^{256}$.

Decap(sk, C): Parse the ciphertext C as (C_1, C_2), and proceed as follows:

1) Compute $\delta \leftarrow$ EMBLEM.CPA.Decrypt(sk, C_1).
2) Compute $r = G(\delta)$.
3) Compute $e \leftarrow$ EMBLEM.CPA.Encrypt($pk, \delta; r$) and $d = \hat{H}(\delta)$.
   - If $e \neq C_1$ or $d \neq C_2$, output $\perp$.
4) Otherwise, output $K = H(\delta, C_1, C_2)$.

Note that all the ephemeral values selected in the EMBLEM .CPA.Encrypt algorithm are determined by the random r. The correctness of this scheme is derived from that of the underlying CPA-secure PKE scheme [19].

Theorem 3.3 (Correctness). *If EMBLEM.CPA is $(1-\epsilon)$-correct, then EMBLEM is $(1-\epsilon)$-correct in the quantum random oracle model.*

EMBLEM is tightly IND-CCA secure in the (classical) random oracle model (by Theorem 3.4) and is non-tightly IND-CCA secure in the quantum random oracle model (by Theorem 3.5). Note that the hash function $\hat{H}$ is not required to prove IND-CCA security in the (classical) random oracle model.

Theorem 3.4 (Theorems 2 and 3 in [19]). *It is assumed that EMBLEM.CPA is δ-correct. For any IND-CCA adversary $\mathcal{B}$ issuing at most q_D decryption queries, at most q_G queries to the random oracle G, and at most q_H queries to the random oracle H, there exists an IND-CPA adversary $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{EMBLEM}}(\mathcal{B}) \leq q_H \cdot \delta + \tfrac{q_H + 2q_G + 1}{2^{256}} + 3 \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{EMBLEM.CPA}}(\mathcal{A})$$

and the running time of $\mathcal{A}$ is approximately that of $\mathcal{B}$.

Theorem 3.5 (Theorems 8 and 9 in [19]). *It is assumed that EMBLEM.CPA is δ-correct. For any IND-CCA quantum adversary $\mathcal{B}$ issuing at most q_D (classical) decryption queries, at most q_G queries to the quantum random oracle G, at most q_H queries to the quantum random oracle H, and at most $q_{\hat{H}}$ queries to the quantum random oracle $\hat{H}$, there exists an IND-CPA quantum adversary $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{IND-CCA}}_{\mathsf{EMBLEM}}(\mathcal{B}) \leq (2q_H + q_{G'}) \cdot \\ \sqrt{8 \cdot \delta(q_G+1)^2 + (1+2q_G)\sqrt{\mathsf{Adv}^{\text{IND-CPA}}_{\mathsf{EMBLEM.CPA}}(\mathcal{A})}}$$

and the running time of $\mathcal{A}$ is approximately that of $\mathcal{B}$.

IV. Ring Variant of EMBLEM

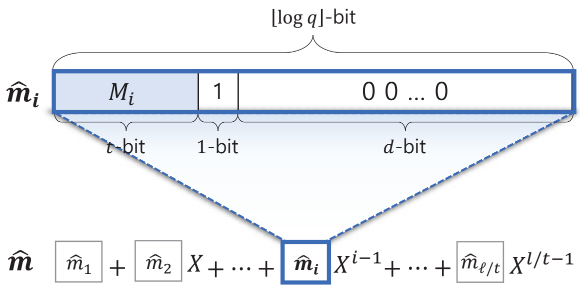
In this section, we propose a ring variant of EMBLEM, called R.EMBLEM. An IND-CPA secure PKE scheme over rings (R.EMBLEM.CPA is first constructed in Section IV-A), and it is converted into an IND-CCA secure KEM (R.EMBLEM in Section IV-B) using the KEM variant of the FO transformation [19].

A. *R.EMBLEM.CPA (CPA-Secure PKE over Rings)*

• **Encoding and decoding functions over rings.** Let $\mathcal{M} = \{0,1\}^{\ell}$ be the message space. The encoding function `R.encode`, which takes as input a bit string and outputs a polynomial, and its inverse function `R.decode` are defined. `R.encode` operates similarly to the `encode` function in Section III-A, except that the output is in polynomial form over rings. For an ℓ-bit message M, a block size t, an upper bound d for the error size, and a modulus q, **R.encode** and **R.decode** operate as follows:

▶ **R.encode**(M, t, d, q)

1) Split the message into t-bit blocks (it is assumed that t divides ℓ) and generate message blocks $\{M_i\}$ for $i \in [1, \ell/t]$.
2) Output a polynomial $\hat{m}$ whose i-th coefficient is computed as $\hat{m}_i \leftarrow M_i||1||0^d$ for $i \in [1, \ell/t]$ (as in Fig. 4).



⌊log q⌋-bit
$\hat{m}_i$
M_i
1
0 0 ... 0
t-bit
1-bit
d-bit
$\hat{m}$
$\hat{m}_1 + \hat{m}_2 X + \cdots + \hat{m}_i X^{i-1} + \cdots + \hat{m}_{\ell/t} X^{l/t-1}$

Fig. 4: Encoding function **R.encode**

▶ **R.decode**$(\hat{m}, t, q)$

1) Compute $M_i = [\hat{m}_i]_t$ for $i \in [1, l/t]$, where $\hat{m}_i$ is the i-th coefficient of $\hat{m}$.
2) Output ℓ-bit string $M = M_1|| \cdots ||M_{l/t}$.

For the cyclotomic polynomial $f(x) = x^n + 1 \in \mathbb{Z}[x]$, let $R_q = \mathbb{Z}_q[x]/\langle f(x)\rangle$ be the ring of integer polynomials modulo both $f(x)$ and q. An element of R_q can be represented by a polynomial of degree less than n whose coefficients are in $\mathbb{Z}_q$.

• **Truncate function.** To improve transmission efficiency, a function Trunc is used that truncates the ciphertext [49]. Let $a \in R_q$ be a polynomial of degree n, represented by $a = a_1 + a_2X + \cdots + a_nX^{n-1}$. For $1 \leq \ell \leq n$, Trunc is defined as follows:

$$\mathsf{Trunc}(a, \ell) = a_1 + a_2X + \cdots + a_\ell X^{\ell-1}.$$

• **Scheme.** Given a security parameter 1^λ, the system-wide parameter $params$ is generated as follows: Positive integers n, t and the modulus q are chosen. A standard deviation $\sigma = s/\sqrt{2\pi}$ for $\mathcal{GD}_s$ and a positive integer $B < \sigma$ are chosen. Let R_q denote the reduction of a ring R modulo q, i.e., $R_q = \mathbb{Z}_q[X]/\langle f(x)\rangle$. Then, $params$ is given by $(n, t, q, B, R_q, \mathcal{GD}_s)$. It is assumed that $params$ is used for all algorithms in R.EMBLEM.CPA.

KeyGen(1^λ): Choose a random polynomial $a \leftarrow R_q$, a secret polynomial $x \in R_q$ where the coefficient vector is sampled randomly from $[-B, B]^n$. [3] Choose an error polynomial $e \in R_q$, where the coefficient of e is sampled randomly from $\mathcal{GD}_s^n$. Compute $b = a \cdot x + e \in R_q$. The key pair (pk, sk) is given by $pk = (a, b)$ and $sk = (x)$.

Encrypt(pk, M): $\ell = 256 \leq n$. To encrypt a message, proceed as follows:

1) $\hat{m} \leftarrow$ **R.encode**(M, t, d, q).
2) Choose a random $z \in \{0,1\}^{256}$.
3) $(r, e_1, e_2) \leftarrow$ **Sam**(z), where r and (e_1, e_2) are polynomials whose coefficients are sampled from $[-B, B]$ and $\mathcal{GD}_s$, respectively.
4) Compute $c_1 = r \cdot a + e_1$ and $c_2 \leftarrow \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m}$.
5) Return the ciphertext $c = (c_1, c_2) \in R_q^2$.

Decrypt(sk, c): Parse the ciphertext c as (c_1, c_2).

1) Compute $d \leftarrow \mathsf{Trunc}(c_1 \cdot x,\ \ell/t)$.
2) Compute $\hat{m} = c_2 - d$.
3) Output $M \leftarrow$ **R.decode**$(\hat{m}, t, q)$.

Theorem 4.1 (Correctness). *Let $(r \cdot e)^{(i)}$, $e_2{}^{(i)}$, and $(e_1 \cdot x)^{(i)}$ be the i-th coefficients of the polynomials $(r \cdot e), e_2$, and $(e_1 \cdot x)$. Let $\epsilon = \Pr\left[\max_{i\in[1,\ell/t]} \left\{|(r \cdot e)^{(i)}| + |e_2^{(i)}| + |(e_1 \cdot x)^{(i)}|\right\} \geq 2^d\right]$, where $d = \lfloor \log q \rfloor - (t+1)$. Then R.EMBLEM.CPA is $(1-\epsilon)$-correct.*

Proof. For the ciphertext c, let (c_1, c_2) be $c_1 = r \cdot a + e_1$, $c_2 \leftarrow \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m}$. In the decryption phase,

$$\begin{aligned} c_2 - d &= \mathsf{Trunc}\big(r \cdot (a \cdot x + e) + e_2,\ \ell/t\big) + \hat{m} \\ &\qquad - \mathsf{Trunc}\big((r \cdot a + e_1) \cdot x,\ \ell/t\big) \\ &= \mathsf{Trunc}(r \cdot e + e_2 - e_1 \cdot x,\ \ell/t) + \hat{m}. \end{aligned}$$

The i-th coefficient of $\hat{m}$ is encoded as $M_i||1||0^d$ for $i \in [1, \ell/t]$, where $M_i \in \{0,1\}^t$ is the i-th message block of M. Therefore, if the absolute size of i-th coefficient in $(re + e_2 - e_1x)$ is less than 2^d, it does not affect the message M_i. As a result, the decryption of R.EMBLEM.CPA performs correctly with probability $1 - \epsilon$. ☐

Theorem 4.2 (Security). R.EMBLEM.CPA *is IND-CPA secure if the* smaRLWE$_{n,q}$ *assumption holds.*

Proof. The proof proceeds with a sequence of games, as in Theorem 3.2. In Game 0, the public key is a small secret ring-LWE instance and the ciphertext is an encryption of M_0. In Game 7, the public key is a small secret ring-LWE instance, and the ciphertext is an encryption of M_1. Let $\mathcal{D}_i$ denote the distribution of Game i. It will be shown that $\mathcal{D}_0$ and $\mathcal{D}_7$ are computationally indistinguishable for the adversary.

▷ **Game 0.** This is the real game, where the ciphertext is an encryption of M_0. Note that $\hat{m}_0 \leftarrow$ **R.encode**(M_0, t, d, q). $\mathcal{D}_0$ is given as follows:

$$\begin{aligned} \mathcal{D}_0 = \{&pk \leftarrow (a, b = a \cdot x + e), \\ &C \leftarrow (c_1 = r \cdot a + e_1, c_2 = \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m}_0)\}. \end{aligned}$$

▷ **Game 1.** In Game 1, the public key b is uniformly generated at random. The remaining part is the same as in Game 0. $\mathcal{D}_1$ is given as follows:

$$\begin{aligned} \mathcal{D}_1 = \{&pk \leftarrow (a, \boxed{b \leftarrow \mathcal{U}(R_q)}), \\ &C \leftarrow (c_1 = r \cdot a + e_1, c_2 = \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m}_0)\}. \end{aligned}$$

▷ **Game 2.** In Game 2, c_1 is changed into a random polynomial. The remaining part is the same as in Game 1. $\mathcal{D}_2$ is given as follows:

$$\mathcal{D}_2 = \{pk \leftarrow (a, b \leftarrow \mathcal{U}(R_q)),$$

[3]As in EMBLEM.CPA, the coefficients of x can be generated using PRF, and thus it is possible to store only a seed.

$$C \leftarrow (\boxed{c_1 = u_1}, c_2 = \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m_0})\}.$$

▷ **Game 3.** In Game 3, the (truncated) c_2 is changed into a random polynomial. The remaining part is the same as in Game 2. $\mathcal{D}_3$ is given as follows:

$$\mathcal{D}_3 = \{pk \leftarrow (a, b \leftarrow \mathcal{U}(R_q)), \\ C \leftarrow (c_1 = u_1, \boxed{c_2 = \mathsf{Trunc}(u_2,\ \ell/t)} + \hat{m_0})\}.$$

▷ **Game 4.** In Game 4, the message M_0 is changed into M_1. Note that $\hat{m_1} \leftarrow \mathbf{R.decode}(M_1, t, q)$. The remaining part is the same as in Game 3. $\mathcal{D}_4$ is given as follows:

$$\mathcal{D}_4 = \{pk \leftarrow (a, b \leftarrow \mathcal{U}(R_q)), \\ C \leftarrow (c_1 = u_1, \boxed{c_2 = \mathsf{Trunc}(u_2,\ \ell/t) + \boxed{\hat{m_1}}})\}.$$

▷ **Game 5.** In Game 5, c_2 is restored to the small secret ring-LWE instance, and the remaining part is the same as in Game 4. $\mathcal{D}_5$ is given as follows:

$$\mathcal{D}_5 = \{pk \leftarrow (a, b \leftarrow \mathcal{U}(R_q)), \\ C \leftarrow (c_1 = u_1, \boxed{c_2 = \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t)} + \hat{m_1})\}.$$

▷ **Game 6.** In Game 6, c_1 is restored to the small secret ring-LWE instance, and the remaining part is the same as in Game 5. $\mathcal{D}_6$ is given as follows:

$$\mathcal{D}_6 = \{pk \leftarrow (a, b \leftarrow \mathcal{U}(R_q)), \\ C \leftarrow (\boxed{c_1 = r \cdot a + e_1}, c_2 = \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m_1})\}.$$

▷ **Game 7.** In Game 7, b is restored to the small secret ring-LWE instance, and the remaining part is the same as in Game 6. $\mathcal{D}_7$ is given as follows:

$$\mathcal{D}_7 = \{pk \leftarrow (a, \boxed{b = a \cdot x + e}), \\ C \leftarrow (c_1 = r \cdot a + e_1, c_2 = \mathsf{Trunc}(r \cdot b + e_2,\ \ell/t) + \hat{m_1})\}.$$

$\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable under the $\mathsf{smaRLWE}_{n,q}$ assumption. $\mathcal{D}_1$ and $\mathcal{D}_2$ are also computationally indistinguishable under the $\mathsf{smaRLWE}_{n,q}$ assumption. $\mathcal{D}_2$ and $\mathcal{D}_3$ are computationally indistinguishable under the $\mathsf{smaRLWE}_{n,q}$ assumption. In Games 3 and 4, the ciphertexts are computed in a one-time pad manner, and thus $\mathcal{D}_3$ and $\mathcal{D}_4$ are statistically indistinguishable. The hybrid games, i.e., Games 4–7, proceed as in the case of Games 0–3, but in a reverse manner. If the parameters n and q of smaRLWE are set to be as hard as RLWE, then the security of the ring LWE problem can be reduced to that of R.EMBLEM.CPA. □

B. R.EMBLEM (CCA-Secure KEM over Rings)

In this section, we propose an IND-CCA secure KEM, called R.EMBLEM, in the quantum random oracle model. To achieve CCA security, the KEM variant on the FO transformation [19] is applied to R.EMBLEM.CPA.

Let $\mathcal{M} = \{0,1\}^{256}$ be the message space of R.EMBLEM .CPA. The system parameter $params$ is given as in R.EMBLEM.CPA, except that the hash functions $G, H, \hat{H}$ described in Section III-B are additionally included. R.EMBLEM = (KeyGen, Encap, Decap) is constructed as follows:

KeyGen(1^λ): Same as R.EMBLEM.CPA.KeyGen.

Encap(pk): Let $n > 256$. To generate a key K and a ciphertext c, proceed as follows:

1) Select a random string $\delta \leftarrow \{0,1\}^{256}$ and compute $z = G(\delta)$.
2) Compute $(c_1, c_2) \leftarrow$R.EMBLEM.CPA.Encrypt$(pk, \delta; z)$ and $c_3 = \hat{H}(\delta)$.
3) Compute $K = H(\delta, c_1, c_2, c_3)$.
4) Return the ciphertext $c = (c_1, c_2, c_3) \in R_q^2 \times \{0,1\}^{256}$ and the key $K \in \{0,1\}^{256}$.

Decap(sk, c): Parse the ciphertext c as (c_1, c_2, c_3) and proceed as follows:

1) Compute $\delta \leftarrow$R.EMBLEM.CPA.Decrypt$(sk, (c_1, c_2))$.
2) Compute $z = G(\delta)$.
3) Compute $(d_1, d_2) \leftarrow$R.EMBLEM.CPA.Encrypt$(pk, \delta; z)$ and $d_3 = \hat{H}(\delta)$.
   - If $(d_1, d_2) \neq (c_1, c_2)$ or $d_3 \neq c_3$, output $\perp$.
4) Otherwise, output $K = H(\delta, c_1, c_2, c_3)$.

The correctness of this scheme is derived from that of the underlying CPA-secure PKE scheme.

Theorem 4.3 (Correctness). *If R.EMBLEM.CPA is $(1-\epsilon)$-correct, then R.EMBLEM is $(1-\epsilon)$-correct in the quantum random oracle model.*

As in EMBLEM, R.EMBLEM is tightly IND-CCA secure in the (classical) random oracle model (by Theorem 4.4) and is non-tightly IND-CCA secure in the quantum random oracle model (by Theorem 4.5).

Theorem 4.4 (Theorems 2 and 3 in [19]). *It is assumed that R.EMBLEM.CPA is δ-correct. For any IND-CCA adversary $\mathcal{B}$ issuing at most q_D decryption queries, at most q_G queries to random oracle G, and at most q_H queries to random oracle H, there exists an IND-CPA adversary $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{IND-CCA}}_{\text{R.EMBLEM}}(\mathcal{B}) \leq q_H \cdot \delta + \frac{q_H + 2q_G + 1}{2^{256}} + 3 \cdot \mathsf{Adv}^{\text{IND-CPA}}_{\text{R.EMBLEM.CPA}}(\mathcal{A})$$

and the running time of $\mathcal{A}$ is approximately that of $\mathcal{B}$.

Theorem 4.5 (Theorems 8 and 9 in [19]). *It is assumed that R.EMBLEM.CPA is δ-correct. For any IND-CCA quantum adversary $\mathcal{B}$ issuing at most q_D (classical) decryption queries, at most q_G queries to the quantum random oracle G, at most q_H queries to the quantum random oracle H, and at most $q_{\hat{H}}$ queries to the quantum random oracle $\hat{H}$, there exists an IND-CPA quantum adversary $\mathcal{A}$ such that*

$$\mathsf{Adv}^{\text{IND-CCA}}_{\text{R.EMBLEM}}(\mathcal{B}) \leq (2q_H + q_{G'}) \cdot \\ \sqrt{8 \cdot \delta (q_G + 1)^2 + (1 + 2q_G)\sqrt{\mathsf{Adv}^{\text{IND-CPA}}_{\text{R.EMBLEM.CPA}}(\mathcal{A})}}$$

and the running time of $\mathcal{A}$ is approximately that of $\mathcal{B}$.

V. Parameter Selection

A. Distributions

• **Secret key distribution $\mathcal{D}_\mathbf{s}$.** The distribution $\mathcal{D}_s$ is uniform distribution over $[-B, B]$ for a positive integer B. As the value B decreases, the expected size of the decryption error can also decreases. Three cases are considered, where B is set to be 1, 2, and 3.

• **Error distribution $\mathcal{D}_\mathbf{e}$.** The distribution $\mathcal{D}_e$ is the discrete Gaussian distribution with the Gaussian parameter s, i.e., $\mathcal{GD}_s$. For the standard deviation $\sigma = s/\sqrt{2\pi}$ of $\mathcal{GD}_s$, σ is set to 25 for EMBLEM and 3 for R.EMBLEM.

• **Message space $\mathcal{M}$.** $\mathcal{M}$ is set to $\{0,1\}^{256}$ (i.e., $\ell = 256$). In EMBLEM, k, t, v are set to be positive integers such that $k \times t \times v = \ell = 256$. A message $m \in \{0,1\}^{256}$ is encoded as a $v \times k$ matrix, where each element is associated with a t-bit message block.

B. Probability of Decryption Failure

For simplicity, the case $v = 1$ is considered in EMBLEM. Then, $\mathbf{R} \in [-B, B]^m$, $\mathbf{E_1} \in \mathcal{GD}_s^n$, and $\mathbf{E_2} \in \mathcal{GD}_s^k$. For $i \in [1, k]$, let

- $\mathbf{E} = \{E^{(i)}\}$, where $E^{(i)} \in \mathcal{GD}_s^m$ is the i-th column of $\mathbf{E}$.
- $\mathbf{E_2} = \{e_2^{(i)}\}$, where $e_2^{(i)} \in \mathcal{GD}_s$ is the i-th entry of $\mathbf{E_2}$.
- $\mathbf{X} = \{X^{(i)}\}$, where $X^{(i)} \in [-B, B]^n$ is the i-th column of $\mathbf{X}$.

To ensure the ***correctness*** of EMBLEM, the size of the decryption error d should be set so as to satisfy the following equation:

$$\Pr\Big[\max_{i \in [1,k]} \Big\{|\langle \mathbf{R}, E^{(i)}\rangle| + |e_2^{(i)}| + |\langle \mathbf{E_1}, X^{(i)}\rangle|\Big\} < 2^d\Big] = 1 - \epsilon. \tag{3}$$

Once d is determined, for a given t, the size of the modulus q satisfying $\lfloor \log q \rfloor = t + 1 + d$ should be determined. In the parameter setting (in Section VII), d was set so that the probability ϵ of decryption failure is negligible, namely, $\epsilon \approx 2^{-140}$, using the following Lemmas:

Lemma 5.1 (Lemma 2.4 of [50]). *For any real $s > 0$ and $Q > 0$, and any $x \in \mathbb{R}^n$, we have*

$$\Pr\big[\ |\langle x, \mathcal{GD}_{\mathbb{Z}^n,s}\rangle| \geq Q \cdot s||x||\ \big] < 2e^{-\pi \cdot Q^2}, \tag{4}$$

where the standard deviation σ of $\mathcal{GD}_{\mathbb{Z}^n,s}$ is set to $\sigma = s/\sqrt{2\pi}$.

Lemma 5.2 (Lemma 3.3 of [51]). *For any $r > 0$ and $z \leftarrow \mathcal{GD}_s$ of the standard deviation $\sigma = s/\sqrt{2\pi}$, we have*

$$\Pr\big[|z| > T\sigma\big] < 2e^{-T^2/2}. \tag{5}$$

By Lemma 5.1, $|\langle \mathbf{R}, E^{(i)}\rangle|$ and $|\langle \mathbf{E_1}, X^{(i)}\rangle|$ can be determined with probability $2e^{-\pi \cdot Q^2}$ as follows:

- $|\langle \mathbf{R}, E^{(i)}\rangle| \leq Q \cdot \sigma\sqrt{2\pi} \cdot ||\mathbf{R}||$.
- $|\langle \mathbf{E_1}, X^{(i)}\rangle| \leq Q \cdot \sigma\sqrt{2\pi} \cdot ||X^{(i)}||$.

Here, Q is set to approximately 5.5776, so that the probability $2e^{-\pi \cdot Q^2} \approx 2^{-140}$ in Lemma 5.1 (i.e., $2e^{-\pi \cdot 5.5776^2} \approx 2^{-140}$). By Lemma 5.2, $|e_2^{(i)}|$ can also be estimated with probability $2e^{-T^2/2}$ as follows:

- $|e_2^{(i)}| \leq T\sigma$.

Here, T is set to approximately 13.98, so that the probability $2e^{-T^2/2} \approx 2^{-140}$ in Lemma 5.2 (i.e., $2e^{-13.98^2/2} \approx 2^{-140}$). Therefore, for all $i \in [1, k]$, there is d such that

$$Q \cdot \sigma\sqrt{2\pi} \cdot ||\mathbf{R}|| + T\sigma + Q \cdot \sigma\sqrt{2\pi} \cdot ||X^{(i)}|| < 2^d,$$

where $Q = 5.5776$ and $T = 13.98$.

Case 1 ($B = 1$)

If $\mathbf{R}$ and $X^{(i)}$ are uniformly sampled from $\{-1, 0, 1\}^m$ and $\{-1, 0, 1\}^n$ at random (i.e., $B = 1$), the expected values of $||\mathbf{R}||$ and $||X^{(i)}||$ are $\sqrt{\frac{2}{3}m}$ and $\sqrt{\frac{2}{3}n}$, respectively. If $(m, n, \sigma) = (1008, 824, 25)$, each value can be calculated as follows:

▶ $Q \cdot \sigma\sqrt{2\pi} \cdot ||\mathbf{R}|| = 5.5776 \times 25\sqrt{2\pi} \times \sqrt{\frac{2}{3} \cdot 1008} \approx 2^{13.15}$.

▶ $T\sigma = 13.98 \times 25 \approx 2^{8.45}$.

▶ $Q \cdot \sigma\sqrt{2\pi} \cdot ||X^{(i)}|| = 5.5776 \times 25\sqrt{2\pi} \times \sqrt{\frac{2}{3} \cdot 824} \approx 2^{13}$.

It follows that $|\langle \mathbf{R}, E^{(i)}\rangle| + |e_2^{(i)}| + |\langle \mathbf{E_1}, X^{(i)}\rangle| < 2^{13.15} + 2^{8.45} + 2^{13} \approx 2^{14.11} < 2^d$. Therefore, $d = 15$ is sufficient in this setting.

Case 2 ($B = 2$)

If $\mathbf{R}$ and $X^{(i)}$ are uniformly sampled from $\{-2, -1, 0, 1, 2\}^m$ and $\{-2, -1, 0, 1, 2\}^n$ at random (i.e., $B = 2$), the expected values of $||\mathbf{R}||$ and $||X^{(i)}||$ are $\sqrt{2m}$ and $\sqrt{2n}$, respectively. If $(m, n, \sigma) = (1016, 784, 25)$, each value can be calculated as follows:

▶ $Q \cdot \sigma\sqrt{2\pi} \cdot ||\mathbf{R}|| = 5.5776 \times 25\sqrt{2\pi} \times \sqrt{2 \cdot 1016} \approx 2^{13.94}$.

▶ $T\sigma = 13.98 \times 25 \approx 2^{8.45}$.

▶ $Q \cdot \sigma\sqrt{2\pi} \cdot ||X^{(i)}|| = 5.5776 \times 25\sqrt{2\pi} \times \sqrt{2 \cdot 784} \approx 2^{13.76}$.

It follows that $|\langle \mathbf{R}, E^{(i)}\rangle| + |e_2^{(i)}| + |\langle \mathbf{E_1}, X^{(i)}\rangle| < 2^{13.94} + 2^{8.45} + 2^{13.76} \approx 2^{14.87} < 2^d$. Therefore, $d = 15$ is sufficient in this setting.

Case 3 ($B = 3$)

If $\mathbf{R}$ and $X^{(i)}$ are uniformly sampled from $\{-3, -2, -1, 0, 1, 2, 3\}^m$ and $\{-3, -2, -1, 0, 1, 2, 3\}^n$ at random (i.e., $B = 3$), the expected values of $||\mathbf{R}||$ and $||X^{(i)}||$ are $\sqrt{4m}$ and $\sqrt{4n}$, respectively. If $(m, n, \sigma) = (936, 808, 25)$, each value can be calculated as follows:

▶ $Q \cdot \sigma\sqrt{2\pi} \cdot ||\mathbf{R}|| = 5.5776 \times 25\sqrt{2\pi} \times \sqrt{4 \cdot 936} \approx 2^{14.38}$.

▶ $T\sigma = 13.98 \times 25 \approx 2^{8.45}$.

▶ $Q \cdot \sigma\sqrt{2\pi} \cdot ||X^{(i)}|| = 5.5776 \times 25\sqrt{2\pi} \times \sqrt{4 \cdot 808} \approx 2^{14.28}$.

It follows that $|\langle \mathbf{R}, E^{(i)}\rangle| + |e_2^{(i)}| + |\langle \mathbf{E_1}, X^{(i)}\rangle| < 2^{14.38} + 2^{8.45} + 2^{14.28} \approx 2^{15.34} < 2^d$. Therefore, $d = 16$ is sufficient in this setting.

It should be noticed that the same analysis as above can also be applied to the ring-based constructions R.EMBLEM.CPA and R.EMBLEM.

C. Proposed parameter sets

Table I and II present the parameter settings for EMBLEM and R.EMBLEM, respectively. The dimension n and the number of samples m were set according to Equation (2), and the upper bound d for the decryption error were set based on the above analysis. The parameterization is aimed

(λ = 128)	**FrodoKEM**	**LOTUS**	**EMBLEM**					
			[I]	**[II]**	**[III]**	**[IV]**	**[V]**	**[VI]**
m	640	832	1,008	1,008	1,016	1,016	936	936
n	640	832	824	824	784	784	808	808
k	8	256	64	32	64	32	64	32
$\log q$	15	13	20	20	20	20	21	21
σ	2.75	3	25	25	25	25	25	25
t	4	1	4	4	4	4	4	4
v	8	1	1	2	1	2	1	2
PK size (bytes)	9,616	658,950	161,312	80,672	162,592	81,312	78,656	39,344
SK size (bytes)	10,256	700,420	32	32	32	32	32	32
CT size (bytes)	9,736	1,144	2,252	4,312	2,152	4,112	4,442	8,684
KeyGen (ms)	0.461	16.393	29.059	19.361	26.053	16.809	27.483	17.502
Encap (ms)	0.631	0.151	0.750	1.085	0.563	0.952	0.458	0.797
Decap (ms)	0.654	0.179	0.704	0.908	0.546	0.723	0.444	0.625
Prob. decryption failure	$2^{-148.8}$	2^{-256}	2^{-140}	2^{-140}	2^{-140}	2^{-140}	2^{-140}	2^{-140}
Security level	128	128	129.3	129.3	129.6	129.6	129.1	129.1
Secret distribution	$\mathcal{GD}_s$	$\mathcal{GD}_s$	$\mathcal{U}[-1,1]$	$\mathcal{U}[-1,1]$	$\mathcal{U}[-2,2]$	$\mathcal{U}[-2,2]$	$\mathcal{U}[-3,3]$	$\mathcal{U}[-3,3]$
Hardness problem	LWE	LWE	LWE	LWE	LWE	LWE	LWE	LWE

Table I: Comparison of CCA-secure KEMs

at achieving at least 128-bit security level and a probability 2^{-140} of decryption failure. Particularly, to estimate the exact security level for each parameter setting, the sage module [10] [4] was used, which is designed for estimating the concrete hardness of a given LWE (or LWR) instance.

VI. Implementation

The constructions were implemented on an Intel Core i7-6700 (Skylake) at 3.4GHz running Ubuntu 16.04 LTS. All the implementations were written in C and for the compilation, GNU GCC version 5.4.0 was used. In this section, the subroutines of the implementation are described.

A. Secret Key Generation

In EMBLEM, the secret key matrix $\mathbf{X} \in [-B, B]^{n\times k}$ is generated as follows. First, the `randombytes()` function is called to generate octet strings rnd[ ] of length $n \times k$. Subsequently, each string is divided by $2B+1$ and $B+1$ is subtracted from the remainder so that the result is in $[-B, B]$. If $B = 1$, the i-th row and j-th column of $\mathbf{X}$, denoted by $X_{i,j}$, is computed by $X_{i,j} \leftarrow (\mathsf{rnd}[i,j]\%3) - 2$, and the result will be in $[-1, 1]$. In R.EMBLEM, the secret key is a polynomial x of degree N. Thus, the `randombytes()` function is called to generate octet strings rnd[ ] of length N, and the remaining part is the same as above. The i-th coefficient of $\mathbf{X}$, denoted by $\hat{X}_i$, is computed by $\hat{X}_i \leftarrow (\mathsf{rnd}[i]\%(2B+1)) - (B+1)$ for $i \in [0, N-1]$.

B. Discrete Gaussian Sampler

An inversion sampling method is used for instantiating the discrete Gaussian sampler, and the table look-up method is used for resistance against side-channel attacks. For $\sigma = 3$, the cumulative density table (CDT), denoted by T, is created from probability density functions. Nine bits are uniformly sampled at random, corresponding to a uniform random integer $x \in [0, 511]$, and an additional bit to determine the sign of the sampled value. For $\sigma = 25$, 11 bits are uniformly sampled at random, corresponding to a uniform random integer $x \in [0, 2047]$, and an additional bit to determine the sign of the sampled value. The lengths of the CDTs for $\sigma = 3$ and $\sigma = 25$ are 16 and 54, respectively.

[4]https://bitbucket.org/malb/lwe-estimator

Algorithm 1: Discrete Gaussian sampler

```
  Input  : CDT look-up table $T$
  Output: sampled value $S$
1 sign ← rand()&1;
2 return $S \leftarrow (-\mathsf{sign})^{T[\mathtt{rand()\&0x1ff}]} + \mathsf{sign}$
```

C. Sampling Function

The sampling function **Sam** (in EMBLEM) takes a 256-bit string r as input and outputs $\mathbf{R} \in [-B, B]^{m\times v}$ and $(\mathbf{E_1}, \mathbf{E_2}) \in \mathcal{GD}_s^{v\times(n+k)}$. The matrix $\mathbf{R}$ is generated as follows: for simplicity, it is assumed that $B = 1$. First, the hash function BLAKE512 with input r is computed to obtain digest, whose data type is `int`. Let *len* be the number of `int` arrays of digest, and let digest$[i']$ denote the i'-th index of digest. For $i \in [1, m]$ and $j \in [1, v]$, the i-th row and j-th column of $\mathbf{R}$, denoted by $\mathbf{R}_{i,j}$, is computed as follows:

$$\mathbf{R}_{i,j} \leftarrow (\mathsf{digest}[i']\%3) - 2. \qquad (6)$$

Then, digest$[i']$ is updated by division by 3.

$$\mathsf{digest}[i'] \leftarrow \mathsf{digest}[i']/3. \qquad (7)$$

When digest$[i']$ becomes zero, the next index is considered and the process is repeated. When all indexes have been processed, BLAKE512 is computed with input $r+1$ and the process is repeated again. When all $m \times v$ elements of $\mathbf{R}$ are filled, the process is terminated. At the end of the process, the original r is hashed again (using SHA256), and the result is used as a new *seed* to generate the error matrices $\mathbf{E_1}, \mathbf{E_2}$. The

(λ = 128)	**NewHope**	**Kyber**	**R.EMBLEM**			
			[I]	**[II]**	**[III]**	**[IV]**
n	1024	256	1,024	1,024	1,024	1,024
k	1	3	1	1	1	1
$\log q$	14	13	14	16	14	16
$\log p$	-	9	-	-	-	-
σ	$\sqrt{8}$	$\sqrt{2}$	3	3	3	3
t	1	1	1	2	1	2
PK size (bytes)	1,824	1,088	1,824	2,080	1,824	2,080
SK size (bytes)	3,680	2,400	32	32	32	32
CT size (bytes)	2,208	1,152	2,272	2,336	2,272	2,336
KeyGen (ms)	0.432	0.315	0.124	0.307	0.122	0.295
Encap (ms)	0.635	0.419	0.152	0.276	0.149	0.272
Decap (ms)	0.721	0.512	0.217	0.492	0.219	0.417
Prob. decryption failure	2^{-216}	2^{-142}	2^{-140}	2^{-140}	2^{-140}	2^{-140}
Security level	233	161	219.2	181.3	236.6	193.5
Secret distribution	$\mathbb{Z}_q$	$\mathcal{B}_\eta$	$\mathcal{U}[-1,1]$	$\mathcal{U}[-1,1]$	$\mathcal{U}[-2,2]$	$\mathcal{U}[-2,2]$
Hardness problem	Ring-LWE	Module-LWE	Ring-LWE	Ring-LWE	Ring-LWE	Ring-LWE

Table II: Comparison of CCA-secure KEMs over Rings

pseudocode for generating the matrix $\mathbf{R}$ and the *seed* is given in Algorithm 2. $\mathbf{E_1}$ and $\mathbf{E_2}$ are generated using a discrete Gaussian sampler with *seed* as input. In Algorithm 1, *seed* is additionally used as a fixed input to the `rand()` function, which allows obtaining the same output $\mathbf{E_1}$ and $\mathbf{E_2}$ for the same *seed*.

Algorithm 2: Generating the matrix $\mathbf{R}$ and the *seed*

```
Input  : 256-bit string $r$
Output: matrix $\mathbf{R} \in [-1,1]^{m \times v}$, seed
1  $cnt \leftarrow 0$;
2  $d \leftarrow r$;
3  while $cnt < m \times v$ do
4      digest ← BLAKE512($d$)
5      for $i = 0$ to ($len$-1) do
6          while digest[$i$] ! = 0 do
7              $\mathbf{R}[cnt]$ ← (digest[$i$]%3) - 2;
8              digest[$i$] ← digest[$i$]/3;
9              $cnt$++;
10         end
11     end
12     $d[0]$++;
13 end
14 seed ← SHA256($r$);
15 return ($\mathbf{R}$, seed)
```

In R.EMBLEM, the sampling function takes a 256-bit string z as an input and outputs the polynomials r, e_1, and e_2. The (coefficients of) polynomial r is generated as in the case of matrix $\mathbf{R}$ in Algorithm 2, except that it is of size n. Similarly, the (coefficients of) polynomials e_1 and e_2 are generated using a discrete Gaussian sampler in Algorithm 1. This sampler generates deterministic outputs for the same input seed z.

D. *Number Theoretic Transform*

The Number Theoretic Transform (NTT) is a specialized version of the discrete Fourier transform in the finite field $F = \mathbb{Z}_p$, i.e., integers modulo the prime p, or a ring. In R.EMBLEM, NTT is based on the Cooly–Tukey butterfly, and the inverse NTT (INTT) is based on the Gentleman–Sande butterfly. Let ψ be a primitive $2n$-th root of unity, $\psi^{1024} \equiv 1 \mod q$, such that $\psi^2 = w$, where $w^{512} \equiv 1 \mod q$. We generate the ψ and ψ^{-1} tables in bit reversed order. For $p = 12289$, $\psi = 1987$ was used, and for $p = 40961$, $\psi = 1044$ was used. Note that the NTT and INTT tables are known to public.

VII. Performance Analysis

A. *Comparison of CCA-secure KEMs*

In Table I, the proposed CCA-secure KEM (EMBLEM) is compared with previous LWE-based KEMs [20], [21]. FrodoKEM [11] was first proposed as a key exchange protocol but later transformed into a CCA-secure KEM [20]. The performance of FrodoKEM is measured using the parameter set `FrodoKEM-640-cSHAKE` [5]. LOTUS [21] is another LWE-based KEM that focuses on reducing the ciphertext size (ignoring the public key size). The performance of LOTUS is measured using the parameter set `lotus-params256 (AVX2)` [6]. Both parameter sets provide 128-bit security level, and their secret keys are chosen from discrete Gaussian distributions.

The six parameter sets for EMBLEM are used in Table I; they are classified into three subsets according to the secret key distribution $[-B, B]^{n \times k}$ for $B = 1, 2, 3$. As B increases, the dimension n and the number m of relevant LWE samples can be reduced (while the 128-bit security level is preserved), which allows reducing the size of the public key. Since v relates to the number of parallel runs, a smaller v in the same parameters m and n leads to a larger public key but simultaneously to shorter ciphertext. Table I shows various trade-offs between the public key size and the ciphertext

[5]https://github.com/Microsoft/PQCrypto-LWEKE
[6]https://www2.nict.go.jp/security/lotus/impl.html

size among the parameter values m, n, v, and B. In all the parameter settings for EMBLEM, the standard deviation σ of the discrete Gaussian distribution is set to $\sigma = 25$, which is considerably larger than in FrodoKEM and LOTUS, and the size of secret key is only 32 bytes regardless of the other parameters. Since the Decap algorithm in EMBLEM should expand the 32-byte secret key to the corresponding secret key matrix, the Decap time was always measured including such an expansion time.

B. Comparison of CCA-secure KEMs over Rings

R.EMBLEM is compared with previous ring-LWE based KEMs [22], [38] in Table II. NewHope [23] was first introduced as a key exchange protocol but later transformed into a CCA-secure KEM [22]. NewHope was implemented under the parameter set `NewHope-1024-CCA-KEM` [7] at 128-bit security level. Kyber [38] was designed based on the module-LWE problem with $k = 3$, which implies that three polynomials (from an underlying ring) are used as a secret key to form a matrix-type of module-LWE. Kyber was implemented under the parameter set `Kyber-768` [8], where secret and error polynomials are sampled from the centered binomial distribution, denoted by $\mathcal{B}_\eta$, instead of the discrete Gaussian distribution. According to [37], $\mathcal{B}_\eta$ can be replaced with a rounded Gaussian distribution of standard deviation $\sqrt{\eta/2}$. Since $\eta = 4$ in `Kyber-768`, $\mathcal{B}_\eta$ was replaced with a rounded Gaussian distribution with $\sigma = \sqrt{2}$ for comparison.

The four parameter sets for R.EMBLEM are considered in Table II; they are divided into two subsets according to $B = 1$ or 2. Table II shows that there is no benefit in increasing the value B in R.EMBLEM because all efficiency and security factors in cases [II] and [IV] are rather worse than those in cases [I] and [III]. When cases [I] and [III] are compared, all efficiency factors are nearly identical, except that the security level of [II] is higher than that of [I]. This is because even though B increases from 1 to 2, it is not easy to change the dimension n because of the highly efficient NTT that is optimized for 2^u-type dimensions (where $u \in \mathbb{N}$). Compared to NewHope and Kyber, R.EMBLEM in cases [I] and [III] is faster (in the present experimental environment), and yields the shortest secret keys, which are, additionally, of constant size.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*. Ieee, 1994, pp. 124–134.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE transactions on Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

[4] L. Chen, L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone, *Report on post-quantum cryptography*. US Department of Commerce, National Institute of Standards and Technology, 2016.

[7]https://newhopecrypto.org/software.shtml

[8]https://github.com/pq-crystals/kyber

[5] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*. ACM, 1996, pp. 99–108.

[6] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *In STOC*, 2005.

[7] R. Lindner and C. Peikert, "Better key sizes (and attacks) for lwe-based encryption." in *CT-RSA*, vol. 6558. Springer, 2011, pp. 319–339.

[8] C. Peikert, V. Vaikuntanathan, and B. Waters, "A framework for efficient and composable oblivious transfer." in *Crypto*, vol. 5157. Springer, 2008, pp. 554–571.

[9] S. Bai and S. D. Galbraith, "Lattice decoding attacks on binary lwe," in *Australasian Conference on Information Security and Privacy*. Springer, 2014, pp. 322–337.

[10] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *Journal of Mathematical Cryptology*, vol. 9, no. 3, pp. 169–203, 2015.

[11] J. Bos, C. Costello, L. Ducas, I. Mironov, M. Naehrig, V. Nikolaenko, A. Raghunathan, and D. Stebila, "Frodo: Take off the ring! practical, quantum-secure key exchange from lwe," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1006–1018.

[12] J. H. Cheon, K. Han, J. Kim, C. Lee, and Y. Son, "A practical post-quantum public-key cryptosystem based on spLWE," in *International Conference on Information Security and Cryptology*. Springer, 2016, pp. 51–74.

[13] J.-P. DAnvers, A. Karmakar, S. S. Roy, and F. Vercauteren, "Saber: Module-lwr based key exchange, cpa-secure encryption and cca-secure kem," in *International Conference on Cryptology in Africa*. Springer, 2018, pp. 282–305.

[14] S. Bhattacharya, O. Garcia-Morchon, T. Laarhoven, R. Rietman, M.-J. O. Saarinen, L. Tolhuizen, and Z. Zhang, "Round5: Compact and fast post-quantum public-key encryption," *Submitted for publication, August*, 2018.

[15] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the aes circuit," in *Advances in cryptology–crypto 2012*. Springer, 2012, pp. 850–867.

[16] S. Halevi and V. Shoup, "Algorithms in helib," in *International Cryptology Conference*. Springer, 2014, pp. 554–571.

[17] M. R. Albrecht, "On dual lattice attacks against small-secret lwe and parameter choices in helib and seal," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2017, pp. 103–129.

[18] M. R. Albrecht, B. R. Curtis, A. Deo, A. Davidson, R. Player, E. W. Postlethwaite, F. Virdia, and T. Wunderer, "Estimate all the {LWE, NTRU} schemes!" in *Security and Cryptography for Networks – SCN 2018*, 2018, pp. 351–367.

[19] D. Hofheinz, K. Hövelmanns, and E. Kiltz, "A modular analysis of the fujisaki-okamoto transformation," in *Theory of Cryptography Conference*. Springer, 2017, pp. 341–371.

[20] E. Alkim, J. Bos, L. Ducas, P. Longa, I. Mironov, M. Naehrig, V. Nikolaenko, C. Peikert, A. Raghunathan, D. Stebila, K. Easterbrook, and B. LaMacchia, "Frodokem–learning with errors key encapsulation," 2017. [Online]. Available: https://frodokem.org/files/FrodoKEM-specification-20171130.pdf

[21] L. T. Phong, T. Hayashi, Y. Aono, and S. Moriai, "Lotus: Algorithm specifications and supporting documentation," 2017. [Online]. Available: https://www2.nict.go.jp/security/lotus/LOTUS_specifications.pdf

[22] E. Alkim, R. Avanzi, J. Bos, L. Ducas, A. Piedra, T. Pöppelmann, P. Schwabe, and D. Stebila, "Newhope–algorithm specifications and supporting documentation," 2017. [Online]. Available: https://newhopecrypto.org/data/NewHope_2017_12_21.pdf

[23] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange-a new hope." in *USENIX Security Symposium*, 2016, pp. 327–343.

[24] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM (JACM)*, vol. 56, no. 6, p. 34, 2009.

[25] R. Impagliazzo and D. Zuckerman, "How to recycle random bits," in *Foundations of Computer Science, 1989., 30th Annual Symposium on*. IEEE, 1989, pp. 248–253.

[26] J. Ding, X. Xie, and X. Lin, "A simple provably secure key exchange scheme based on the learning with errors problem." *IACR Cryptology EPrint Archive*, vol. 2012, p. 688, 2012.

[27] C. Peikert, "Lattice cryptography for the internet," in *International Workshop on Post-Quantum Cryptography*. Springer, 2014, pp. 197–219.

[28] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 1–23.

[29] Y. Elias, K. E. Lauter, E. Ozman, and K. E. Stange, "Provably weak instances of ring-lwe," in *Annual Cryptology Conference*. Springer, 2015, pp. 63–92.

[30] W. Castryck, I. Iliashenko, and F. Vercauteren, "Provably weak instances of ring-lwe revisited," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2016, pp. 147–167.

[31] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom, "Flush, gauss, and reload–a cache attack on the bliss lattice-based signature scheme," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2016, pp. 323–345.

[32] C. Peikert, "How (not) to instantiate ring-lwe," in *International Conference on Security and Cryptography for Networks*. Springer, 2016, pp. 411–430.

[33] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from ring-lwe and security for key dependent messages," in *Annual cryptology conference*. Springer, 2011, pp. 505–524.

[34] V. Lyubashevsky, C. Peikert, and O. Regev, "A toolkit for ring-lwe cryptography," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 35–54.

[35] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila, "Post-quantum key exchange for the tls protocol from the ring learning with errors problem," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 553–570.

[36] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, p. 13, 2014.

[37] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," *Designs, Codes and Cryptography*, vol. 75, no. 3, pp. 565–599, 2015.

[38] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "Crystals-kyber: a cca-secure module-lattice-based kem," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2018.

[39] M. Hamburg, "Module-lwe key exchange and encryption: The three bears," 2017.

[40] R. Steinfeld, A. Sakzad, and R. K. Zhao, "Titanium: Proposal for a nist post-quantum public-key encryption and kem standard," 2017.

[41] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," *Advances in Cryptology–EUROCRYPT 2012*, pp. 719–737, 2012.

[42] J. Alwen, S. Krenn, K. Pietrzak, and D. Wichs, "Learning with rounding, revisited," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 57–74.

[43] A. Bogdanov, S. Guo, D. Masny, S. Richelson, and A. Rosen, "On the hardness of learning with rounding over small modulus," in *Theory of Cryptography Conference*. Springer, 2016, pp. 209–224.

[44] J. H. Cheon, D. Kim, J. Lee, and Y. Song, "Lizard: Cut off the tail! a practical post-quantum public-key encryption from lwe and lwr," in *International Conference on Security and Cryptography for Networks*. Springer, 2018, pp. 160–177.

[45] D. Micciancio and C. Peikert, "Hardness of sis and lwe with small parameters," in *Advances in Cryptology–CRYPTO 2013*. Springer, 2013, pp. 21–39.

[46] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehlé, "Classical hardness of learning with errors," in *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*. ACM, 2013, pp. 575–584.

[47] S. Goldwasser, Y. T. Kalai, C. Peikert, and V. Vaikuntanathan, "Robustness of the learning with errors assumption," 2010.

[48] E. E. Targhi and D. Unruh, "Post-quantum security of the fujisaki-okamoto and oaep transforms," in *Theory of Cryptography Conference*. Springer, 2016, pp. 192–216.

[49] M. R. Albrecht, E. Orsini, K. G. Paterson, G. Peer, and N. P. Smart, "Tightly secure ring-lwe based key encapsulation with short ciphertexts," in *Computer Security - ESORICS 2017, Part I*. Springer, 2017, pp. 29–46.

[50] W. Banaszczyk, "Inequalities for convex bodies and polar reciprocal lattices in r n," *Discrete & Computational Geometry*, vol. 13, no. 1, pp. 217–231, 1995.

[51] V. Lyubashevsky, "Digital signatures based on the hardness of ideal lattice problems in all rings," in *Advances in Cryptology–ASIACRYPT 2016: 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II 22*. Springer, 2016, pp. 196–214.