IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 18, 2018, accepted November 25, 2018, date of publication December 3, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2884084

RLizard: Post-Quantum Key Encapsulation Mechanism for IoT Devices

JOOHEE LEE[1], DUHYEONG KIM[1], HYUNGKYU LEE[2], YOUNHO LEE[3], AND JUNG HEE CHEON[1]

[1]Department of Mathematical Sciences, Seoul National University, Seoul 08826, South Korea
[2]Department of Computer Sciences, KAIST, Daejeon 34141, South Korea
[3]ITM Programme, Department of Industrial and Systems Engineering, SeoulTech, Seoul 01811, South Korea

Corresponding authors: Younho Lee (younholee@seoultech.ac.kr) and Jung Hee Cheon (jhcheon@snu.ac.kr)

This work was supported in part by the Samsung Research Funding Center, Samsung Electronics, under Project SRFC-TB1403-52, in part by the Institute for Information and Communications Technology Promotion through the Korea Government (MSIT) under Grant 2017-0-00616 (development of lattice-based post-quantum public-key cryptographic schemes), and in part by the National Research Foundation of Korea thorough the Korea Government (MSIP) under Grant NRF-2016R1C1B2011022.

ABSTRACT We propose the RLizard key encapsulation mechanism (KEM), whose security depends on the ring learning with errors and ring learning with rounding problems. Because RLizard operates on a special type of ring, it is more efficient in terms of both the clock cycles required for key generation and the key size compared with the original Lizard scheme. To demonstrate the superiority of the proposed method over other well-known KEMs, we compared their performances in the 32-bit ARM Internet of Things (IoT) environment. The performance analysis showed that the RLizard KEM requires the fewest clock cycles for key generation, encapsulation, and decapsulation when the parameters are set to support a security level comparable with that of AES-128. In summary, the RLizard KEM is expected to be used for secure communication and authentication between IoT endpoint devices, whose computational power is generally limited.

INDEX TERMS Key encapsulation mechanism, post-quantum cryptography, Internet of Things, security.

I. INTRODUCTION

We live in the age of the Internet of Things (IoT). According to Statistica, as of 2018, 23.14 billion devices around the world are connected to the IoT, and this number is expected to increase to 75.44 billion by 2025 [1]. The IoT can be characterized by communication with minimum human intervention between uniquely identified virtual objects that are associated with some physical objects, including sensors, from which environmental and contextual information can be acquired; self-configured networks, over which such communication occurs; intelligent behavior based on automatic decisions made according to the information obtained by the cooperation of numerous sensors; and new types of applications working on top of them [2].

The security of IoT-based systems and services is a critical issue that requires careful consideration. Compared to conventional Internet-connected devices, less computational power is available to the majority of participating devices in the IoT. Thus, providing security to IoT systems is a challenge. Previous studies have found vulnerabilities in many existing IoT systems and services owing to the lack of security considerations [3]–[6]. Malicious entities can exploit such vulnerabilities to inflict severe damage on IoT systems. Therefore, securing the IoT is of great significance.

The essential properties required to provide security to IoT systems are confidentiality, integrity, and authenticity among IoT devices. In this regard, a basic security operation is the sharing of random session keys between such devices. However, the development of quantum computers and attacks performed with such computers are expected to render traditional cryptosystems, such as RSA, Diffie-Hellman, and ECC, obsolete in the near future, because the underlying hardness problems of such cryptosystems will be completely broken by quantum attacks [7].

To address the above-mentioned problems, researchers have been studying key encapsulation mechanisms (KEMs) with which random session keys can be shared between two devices, and such mechanisms are secure even against quantum attacks [8]–[15]. Furthermore, in line with this direction, in 2017, NIST launched a project to standardize post-quantum cryptographic algorithms, which involves the standardization of post-quantum KEM (pq-KEM) that

is secure against quantum adversaries. Many schemes have been proposed as candidates for pq-KEM [16].

In spite of such active research on pq-KEM, few studies have investigated pq-KEM in consideration of the IoT environment. In view of the importance of IoT security as discussed above, research on pq-KEM for the IoT environment is crucial. In this paper, we propose a new pq-KEM to support the basic security operation in the IoT environment. The proposed KEM, namely Ring Lizard (abbreviated as RLizard), is a modified version of the original Lizard scheme [12], suited for the IoT environment. As the name of the scheme suggests, it involves special mathematical structures called rings. RLizard is advantageous in terms of the required computational and storage resources. As the key generation phase of the original Lizard scheme requires multiplication of two huge matrices, its computational cost is rather high. By contrast, in the case of RLizard, such expensive computation is replaced by simple multiplication of two polynomials, and also the sizes of both the public key and the secret key are also shrunk compared to the original Lizard scheme.

To evaluate the performance of RLizard in the IoT environment, we compare it with existing pq-KEMs by implementing and testing them in the ARM 32-bit MCU environment, which is known to be widely used for IoT devices. For efficient implementation, we reduced the number of required clock cycles compared to the RLizard implementation submitted to the NIST for standardization [16].

We found that the proposed scheme involves the fewest clock cycles in all operations compared with the other schemes. Specifically, 102∼104 bits were required to provide a similar level of quantum security and the number of clock cycles required for key generation, encapsulation, and decapsulation were reduced by around 30.5%, 40.6%, and 3.8%, respectively, compared to *Kyber* [10], which consumes the fewest clock cycles among the other schemes.

The remainder of this paper is organized as follows. Section II introduces the preliminaries to provide a better understanding of the proposed scheme. Section III reviews the related work. Section IV presents the proposed KEM, i.e., RLizard, including its security proof. Section V discusses the implementation of the proposed scheme and compares its performance with that of other schemes in the IoT environment. Finally, Section VI concludes the paper.

II. PRELIMINARIES

A. NOTATIONS

All logarithms are taken to the base 2 unless otherwise indicated. For a positive integer q, we use $\mathbb{Z} \cap (-q/2, q/2]$ as a representative of $\mathbb{Z}_q$. For a real number r, $\lfloor r \rceil$ denotes the integer nearest to r, rounded up in the case of a tie. We denote vectors in bold, e.g., $\mathbf{a}$, and every vector in this paper is a column vector. The norm $\|\cdot\|$ is always the 2-norm in this paper. We use $x \leftarrow D$ to denote the sampling of x according to the distribution D. The sampling is uniform when D is a finite set. For an integer $n \geq 1$, D^n denotes the product of i.i.d. random variables $D_i \sim D$. We use λ to denote the security parameter throughout the paper: all known valid attacks against the cryptographic scheme under study should take $\Omega(2^\lambda)$ bit operations. Here $f(\lambda) = \Omega(g(\lambda))$ means that $f(\lambda)$ is asymptotically lower bounded by $g(\lambda)$, i.e., there exists some k and λ_0 which satisfies $f(\lambda) \geq k \cdot g(\lambda)$ for any $\lambda \geq \lambda_0$.

For an integer d, let $\Phi_d(X)$ be the d-th cyclotomic polynomial of degree $n = \phi(d)$, where $\phi(\cdot)$ is Euler's totient function which denotes the number of coprime positive integers below the input. We express the cyclotomic ring and its residue ring modulo an integer q as $R = \mathbb{Z}[X]/(\Phi_d(X))$ and $R_q = \mathbb{Z}_q[X]/(\Phi_d(X))$, respectively. Further, we identify the vectors of $\mathbb{Z}_q^n$ with the elements of R_q by $(a_0, ..., a_{n-1}) \mapsto \sum_{i=0}^{n-1} a_i X^i$. For any distribution $\mathcal{D}$ over $\mathbb{Z}_q$, sampling a polynomial $\sum_{i=0}^{n-1} a_i X^i \in R_q$ from D^n implies sampling the coefficient vector $(a_0, ..., a_{n-1})$ from the distribution.

For any real $\sigma > 0$, the discrete Gaussian distribution $\mathcal{DG}_\sigma$ is a probability distribution with support $\mathbb{Z}$ that assigns a probability proportional to $\exp(-\pi x^2/\sigma^2)$ to each $x \in \mathbb{Z}$. Note that the variance of $\mathcal{DG}_\sigma$ is close to $\sigma^2/2\pi$ unless σ is extremely small. For an integer $0 \leq h \leq n$, the distribution $\mathcal{HWT}_n(h)$ samples a vector uniformly from $\{0, \pm 1\}^n$, under the condition that it has exactly h nonzero entries.

B. PUBLIC-KEY ENCRYPTION AND KEY ENCAPSULATION MECHANISM

Public-key encryption (PKE) is an encryption system that allows encryption of a message using a public key, which has a corresponding secret key, and a ciphertext can be decrypted only when the secret key is available. Formally, PKE consists of four algorithms, namely Setup, KeyGen, Enc, and Dec, where $\mathsf{Setup}(1^\lambda)$ outputs a parameter set pp, $\mathsf{KeyGen}(params)$ outputs a pair of public key and secret key $(\mathsf{pk}, \mathsf{sk})$, $\mathsf{Enc}(\mathsf{pk}, m)$ outputs a ciphertext $\mathbf{c}$ of an input plaintext m, and $\mathsf{Dec}(\mathsf{sk}, \mathbf{c})$ outputs a plaintext m. We say that a PKE scheme is secure (indistinguishable) against chosen-plaintext attacks, i.e., it has IND-CPA security, if any adversary $\mathcal{A}$ does not have a non-negligible advantage, which means $\mathsf{Adv}^{\mathsf{CPA}}_{\mathsf{PKE}}(\mathcal{A}) =$

$$\left| \Pr\left[b = b' : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ (m_0, m_1) \leftarrow \mathcal{A}(\mathsf{pk}) \\ b \leftarrow \{0, 1\} \\ \mathbf{c}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b' \leftarrow \mathcal{A}(\mathsf{pk}, \mathbf{c}^*) \end{array} \right] - \frac{1}{2} \right| < \epsilon(\lambda),$$

where $\epsilon(\lambda) > 0$ is a negligible function in λ.

A KEM is a one-round protocol that enables two parties to share an ephemeral key. Specifically, a sender produces a ciphertext of an ephemeral key by using a receiver's public key. The ciphertext cannot be explicitly chosen by the sender. Then, the receiver decrypts the ciphertext and obtains the same ephemeral key. The sender's and receiver's algorithms are called encapsulation and decapsulation, which are denoted by Encaps and Decaps, respectively. Following the context from [55], the entire KEM procedure is described as a tuple of four algorithms.

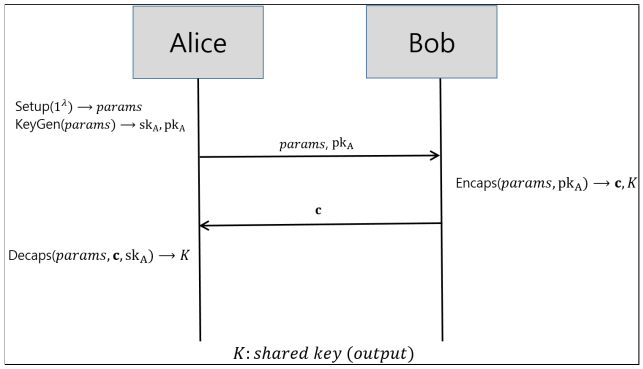

FIGURE 1. KEM protocol.

- $\mathsf{Setup}(1^\lambda)$: Generate a public parameter *params*.
- $\mathsf{KeyGen}(params)$: For an input *params*, generate a public key pk for encapsulation and a secret key sk for decapsulation.
- $\mathsf{Encaps}(\mathsf{pk})$: For input pk, generate and output a ciphertext $\mathbf{c}$ of a key K.
- $\mathsf{Decaps}(\mathsf{sk}, \mathbf{c})$: For input sk and $\mathbf{c}$, output a key K. Note that K can be output as $\perp$, which denotes decapsulation failure.

For the readers' easy understanding, the KEM protocol is depicted below in Fig. 1.

A KEM scheme is said to be secure (indistinguishable) against chosen-ciphertext attacks, i.e., it has IND-CCA security, if any adversary given access to $\mathsf{Decaps}(\mathsf{sk}, \cdot)$ is not able to computationally distinguish the distribution of $(\mathsf{pk}, \mathbf{c}, K)$ and the distribution of $(\mathsf{pk}, \mathbf{c}, K')$, where K' is uniformly randomly chosen from the key space $\mathcal{K}$. In other words, any adversary $\mathcal{A}$ against an IND-CCA secure KEM is not able to have a non-negligible advantage $\mathsf{Adv}^{\mathsf{CCA}}_{\mathsf{KEM}}(\mathcal{A})$ defined as

$$\left| \Pr\left[b = b' : \begin{array}{c} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}) \\ b \leftarrow \{0,1\} \\ (\mathbf{c}^*, K_0^*) \leftarrow \mathsf{Encaps}(\mathsf{pk}) \\ K_1^* \leftarrow \mathcal{K} \\ b' \leftarrow \mathcal{A}^{\mathsf{Decaps}(\mathsf{sk},\cdot)}(\mathsf{pk}, \mathbf{c}^*, K_b^*) \end{array} \right] - \frac{1}{2} \right|.$$

Here, note that the decryption query cannot be applied to the ciphertext $\mathbf{c}^*$ of a given sample.

C. RING LEARNING WITH ERRORS

The ring learning with errors (RLWE) problem, which was proposed by Lyubashevsky *et al.* [17], and by Stehlé *et al.* [18] with a slightly different notion, is a ring variant of the learning with errors (LWE) problem [19]. For positive integers n and q, and an irreducible polynomial $f(X) \in \mathbb{Z}[X]$ of degree n, we define the number field $K := \mathbb{Q}[X]/(f(X))$ and its ring of integers $R := \mathbb{Z}[X]/(f(X))$. We denote by R_q the quotient ring of R modulo q, *i.e.*, $R_q := R/qR$. If $f(X)$ is a cyclotomic polynomial, then it is well known that $R = \mathbb{Z}[X]/(f(X))$ and $R_q = \mathbb{Z}_q[X]/(f(X))$. In this paper, we fix the polynomial $f(X)$ to be the $2n$-th cyclotomic polynomial, where n is a power-of-two integer, *i.e.*, $f(X) = X^n + 1$. Let χ be a distribution over R_q that samples a polynomial with small coefficients compared to q, and let $\mathcal{D}$ be a distribution over R_q. The RLWE problem Ring-$\mathsf{LWE}_{n,q,\chi}(\mathcal{D})$ is to distinguish between the uniform distribution over R_q^2 and the distribution of $(a, a \cdot s + e) \in R_q^2$, where a is uniformly randomly chosen from R_q^2, e is chosen from χ, and s is a secret polynomial sampled from $\mathcal{D}$.

The RLWE problem over the ring R is at least as hard as the approximate shortest vector problem (SVP) and the shortest independent vectors problem (SIVP) on ideal lattices in R [17]. Although we have specified that the polynomial $f(X)$ is a cyclotomic polynomial, it has recently been shown in [20] that the hardness of RLWE is also guaranteed by the hardness assumption of such lattice hard problems on ideal lattices for any irreducible integer polynomial $f(X)$.

In practice, the power-of-2 cyclotomic polynomial $f(X) = X^n + 1$ is typically chosen as a base ring of RLWE due to both the efficiency and the security: the simple form of $f(X) = X^n + 1$ enables very fast polynomial reduction operations, and moreover there exist no known attacks on RLWE over power-of-2 cyclotomic ring [21] while some special algebraic attacks have been proposed in case of non power-of-2 cyclotomic rings [22]–[24].

D. RING LEARNING WITH ROUNDING

The ring learning with rounding (RLWR) problem, proposed by Banerjee *et al.* [25] is a derandomized version of the RLWE problem: instead of adding an error polynomial e to $a \cdot s$, one discards some least significant bits of each coefficient of $a \cdot s$. Formally, it is defined as follows. Let n, q, and $f(X) = X^n + 1$ be defined as in the previous subsection. Instead of error distribution χ, we add a new parameter p, which is called a rounding modulus. We denote the quotient ring of R modulo p by $R_p = R/pR = \mathbb{Z}_p[X]/(X^n + 1)$. For a given $s \in R_q$, we denote by $A^{RLWR}_{n,q,p}(s)$ the distribution of

$$\left(a, \left\lfloor \frac{p}{q} \cdot a \cdot s \right\rceil \right) \in R_q \times R_p,$$

where a is uniformly sampled from R_q. The RLWR problem Ring-$\mathsf{LWR}_{n,q,p}(\mathcal{D})$ is to distinguish the uniform distribution over $R_q \times R_p$ and the distribution $A^{RLWR}_{n,q,p}(s)$, where s is sampled from $\mathcal{D}$. The search version of RLWR, which aims to find s when several samples from $A^{RLWR}_{n,q,p}(s)$ are given, is known to be at least as hard as the search version of RLWE [25], [26].

III. RELATED WORK

A. POST-QUANTUM LATTICE-BASED PKE/KEMS WITH IND-CCA SECURITY

Lattice-based cryptography is one of the most attractive areas of post-quantum cryptography owing to its distinctive advantages of strong security, fast implementation, and versatility. NTRU encryption [27] together with its dedicated padding scheme [28] can be regarded as the earliest example of a lattice-based KEM. Bernstein *et al.* [9] proposed a new variant of NTRU, namely NTRU Prime, by changing the

base ring of NTRU encryption from cyclotomic rings to new rings without some mathematical structures to avoid future analyses, eliminating decryption failure, and developing a constant-time implementation. Recently, Hülsing *et al.* [15] proposed an optimized software implementation for NTRU KEM, which also allows efficient constant-time noise sampling.

Another strategy for constructing a lattice-based IND-CCA pq-KEM is to adopt the LWE problem or its ring or module variants. In this approach, one can first construct IND-CPA PKE and then use well-known CCA conversion, such as Fujisaki-Okamoto conversion [29], along with hash functions to convert it into an IND-CCA pq-KEM. -=The LWE problem was originally adopted to construct a PKE system by Regev [19] in 2005. A drawback of some well-known variants of Regev's PKE scheme [19] is that the required parameters are too large for practical application. Lindner and Peikert (LP) [30] addressed this problem by introducing noise into a combination of LWE samples in the encryption stage. NewHope [8] and Frodo [11] provide efficient KEM instantiations from LP-style encryption. NewHope uses the ring structure and its security is thus based on the Ring-LWE assumption, whereas the security of Frodo is based on the original LWE assumption. Cheon *et al.* [12] proposed an IND-CPA PKE scheme called Lizard as well as its IND-CCA KEM version. Lizard replaces the error sampling process in LP encryption with deterministic rounding; thus, it achieves extremely fast encryption. The security of Lizard is based on both the LWE assumption and the LWR assumption. Kyber [10] is a module-based KEM instantiation of LP-style PKE, and its security is based on the module-LWE assumption. Saber [13] replaces the Gaussian sampling in key generation and encapsulation with a rounding process, and its security is based on the module-LWR assumption.

B. EFFICIENT IMPLEMENTATION OF POST-QUANTUM CRYPTOGRAPHIC PRIMITIVES FOR IOT ENVIRONMENT

Guillen *et al.* [31] implemented NTRUEncrypt [14] in the IoT environment and evaluated its performance. They did not consider the key encapsulation mechanism and focused only on NTRUEncrypt. Boorghany *et al.* [32] implemented NTRUEncrypt and some authentication protocols in the ARM7TDMI and AVR ATmega128 environments, and they evaluated their performance, but they did not discuss the KEM implementation. Liu *et al.* [33] implemented a Ring LWE encryption scheme [17] in the AVR ATmega128 environment in an efficient manner. Later, the same scheme [17] was implemented and its performance was analyzed in the ARM Cortex-A9 and MSP430 environments [34], [35]. However, the KEM performance was not discussed. There are some researches dealing with the KEM for IoT environment [36], [37]. Unfortunately, the underlying hardness problems in their methods are not known to be secure against quantum security.

IV. RLizard KEY ENCAPSULATION MECHANISM

RLizard is a ring version of the Lizard [12] scheme, and it has been submitted to NIST for post-quantum cryptography standardization together with the non-ring version. The security of RLizard is based on the hardness of the Ring-LWE and Ring-LWR problems. RLizard is a rather compact and light version of the Lizard scheme in terms of parameter sizes and key generation speeds, which makes it suitable for IoT implementation. It is infeasible for Lizard to be used for IoT endpoint devices because Lizard requires more than 1000KB storage size to store a key pair in order to support practical level of security: the SRAM sizes for conventional IoT endpoint devices are in the range between a few Kilo Bytes and several hundred Kilo Bytes [12], [38].

In terms of security, IND-CCA security for KEM is adequate for current Internet applications. Furthermore, because establishing a secure channel between entities using a shared key is a basic primitive for all kinds of Internet activities, the efficiency of KEM matters a lot. One of the best known ways to achieve an efficient quantum-safe IND-CCA KEM is to construct a post-quantum IND-CPA PKE first, and then apply a generic Fujisaki-Okamoto conversion which converts any IND-CPA PKEs into IND-CCA KEMs in (quantum) random oracle model. Direct constructions for post-quantum IND-CCA PKE or KEM also exist [39], [40], but they require some blow-up for parameters because of the use of trapdoor one-way functions. Hence, as all other NIST submissions and the original Lizard scheme do, we follow the former approach to achieve an IND-CCA secure KEM so that we first construct IND-CPA PKE, named as RLizard, and then apply a variant of the Fujisaki-Okamoto transformation in [29] to it.

We present our IND-CPA PKE scheme and IND-CCA KEM in Section IV-A and Section IV-B, respectively.

A. BASIC IND-CPA SECURE ENCRYPTION SCHEME

For the simplicity of ring operations, we choose a power-of-two degree in the following description.

- $\mathsf{RLizard.Setup}(1^\lambda)$: Choose positive integers t, p, p', and q such that $t \leq p' \leq p \leq q$ and $p'|p$. Let $n \in \mathbb{Z}$ be a power of 2 and $\Phi(X) = X^n + 1$ be the $2n$-th cyclotomic polynomial. Choose h_s, h_r less than or equal to n, a private key distribution $\mathcal{D}_s$ over R, an ephemeral secret distribution $\mathcal{D}_r$ over R, and a parameter σ for the discrete Gaussian distribution $\mathcal{DG}_\sigma$. Output $params \leftarrow (n, t, q, p, p', \mathcal{D}_s, \mathcal{D}_r, \sigma)$.
- $\mathsf{RLizard.KeyGen}(params)$: Generate a random polynomial $a \leftarrow R_q$. Sample a secret polynomial $s \leftarrow \mathcal{D}_s^n$, and an error polynomial $e \leftarrow \mathcal{DG}_\sigma^n$. Let $b = a \cdot s + e \in R_q$. Output the public key $\mathsf{pk} \leftarrow (a, b) \in R_q^2$ and the secret key $\mathsf{sk} \leftarrow s \in R$.
- $\mathsf{RLizard.Enc}_{\mathsf{pk}}(m)$: For a plaintext $m \in R_t = R/tR$, choose $r \leftarrow \mathcal{D}_r^n$ and compute $c_1' \leftarrow a \cdot r$ and $c_2' \leftarrow b \cdot r$. Output the vector

$$\mathbf{c} \leftarrow (c_1, c_2) \in R_p \times R_p',$$

where $c_1 \leftarrow \lfloor (p/q) \cdot c_1' \rceil \in R_p$ and $c_2 \leftarrow \lfloor (p'/t) \cdot m \rceil + \lfloor (p'/q) \cdot c_2' \rceil \in R_{p'}'$.

- $\mathsf{RLizard.Dec}_{\mathsf{sk}}(\mathbf{c})$: For a ciphertext $\mathbf{c} = (c_1, c_2)$, compute and output the polynomial

$$m' \leftarrow \left\lfloor \frac{t}{p} \left(\frac{p}{p'} \cdot c_2 - c_1 \cdot s \right) \right\rceil \in R_t.$$

Correctness. The correctness condition of the RLizard PKE scheme is presented as follows.

Lemma 1 (Correctness): Assuming that $t \mid p' \mid p \mid q$, the public key encryption RLizard works correctly as long as the following inequality holds for the security parameter λ:

$$\sum_{j=0}^{n-1} \Pr\left[\left| [e \cdot r + s \cdot f]_j \right| \geq \frac{q}{2t} - \frac{q}{2p'} \right] < \mathsf{negl}(\lambda)$$

for $e = \sum_{i=0}^{n-1} e_i X^i$, $r = \sum_{i=0}^{n-1} r_i X^i$, $s = \sum_{i=0}^{n-1} s_i X^i$, and $f = \sum_{i=0}^{n-1} f_i X^i$, where $e_i \leftarrow \mathcal{D}G_\sigma$, $r_i \leftarrow \mathcal{D}_r$, $s_i \leftarrow \mathcal{D}_s$, and $f_i \leftarrow \mathbb{Z}^n_{q/p}$ are independently sampled for all i's and $[\cdot]_j$ denotes the coefficient of X^j.

Proof: Let r be a polynomial sampled from $\mathcal{D}_r^n$ in the encryption procedure, and let $c_1' \leftarrow a \cdot r$ and $c_2' \leftarrow b \cdot r$. The ciphertext is $\mathbf{c} = (c_1 = \lfloor (p/q) \cdot c_1' \rceil, c_2 = \lfloor (p'/t) \cdot m \rceil + \lfloor (p'/q) \cdot c_2' \rceil)$. Let $f \leftarrow c_1' \pmod{q/p} \in R_{q/p}$ and $f' \leftarrow c_2' \pmod{q/p'} \in R_{q/p'}$ so that $(q/p) \cdot c_1 = c_1' - f$ and $(q/p') \cdot c_2 = c_2' - f'$. Note that $f = a \cdot r \pmod{q/p}$ is uniformly and randomly distributed over $R^n_{q/p}$ independently from the choice of r, e, and s. Then, $(p/p') \cdot c_2 - c_1 \cdot s$ is

$$\begin{aligned} &(p/t) \cdot m + (p/q) \cdot (c_2' - s \cdot c_1') - (p/q) \cdot (f' - s \cdot f) \\ &\quad = (p/t) \cdot m + (p/q) \cdot (e \cdot r + s \cdot f) - (p/q) \cdot f'. \end{aligned}$$

Since $f' = (a \cdot s + e) \cdot r \pmod{q/p'}$, an infinite norm of f' is bounded by $q/2p'$. Hence, the correctness of RLizard PKE scheme is guaranteed if, for all $1 \leq j \leq n$, the j-th coefficient of the encryption error is bounded by $p/2t$, or if, $|[e \cdot r + s \cdot f]_j| < q/2t - q/2p'$ for all j with an overwhelming probability, where $[\cdot]_j$ denotes the coefficient of X^j. □

Security. The security of RLizard PKE scheme relies on the hardness of Ring-LWE and Ring-LWR problems. Looking closer, the public key of RLizard can be seen as a Ring-LWE instance with a secret as the secret key, and the encryption of zero is a Ring-LWR instance with the secret as the ephemeral secret polynomial. This implies the following theorem.

Theorem 1 (Security): The PKE scheme RLizard is IND-CPA secure under the hardness assumption of Ring-$\mathsf{LWE}_{n,q,\mathcal{D}G_\sigma}(\mathcal{D}_s^n)$ and Ring-$\mathsf{LWR}_{n,q,p}(\mathcal{D}_r^n)$.

Proof: The security of RLizard encryption scheme is reduced to the simplest case that p' equals to p, since we assume $p' | p$ in our parameter setting. An encryption of m can be generated by adding $(p/t) \cdot m$ to an encryption of zero. Hence, it is enough to show that the pair of public information $\mathsf{pk} = (a, b) \leftarrow \mathsf{RLizard.KeyGen}(params)$ and encryption of zero $\mathbf{c} \leftarrow \mathsf{RLizard.Enc}_{\mathsf{pk}}(0)$ is computationally indistinguishable from the uniform distribution over $R_q^2 \times R_q^2$ for a parameter set $params \leftarrow \mathsf{RLizard.Setup}(1^\lambda)$.

We define the following sequence of distributions:

- $\mathcal{D}_0 = \{(\mathsf{pk}, \mathbf{c}) : \mathsf{pk} \leftarrow \mathsf{RLizard.KeyGen}(params), \mathbf{c} \leftarrow \mathsf{RLizard.Enc}_{\mathsf{pk}}(0)\}$
- $\mathcal{D}_1 = \{(\mathsf{pk}, \mathbf{c}) : \mathsf{pk} \leftarrow R_q^2, \mathbf{c} \leftarrow \mathsf{RLizard.Enc}_{\mathsf{pk}}(0)\}$
- $\mathcal{D}_2 = \{(\mathsf{pk}, \mathbf{c}) : \mathsf{pk} \leftarrow R_q^2, \mathbf{c} \leftarrow R_q^2\}$

The public key $\mathsf{pk} = (a, b) \leftarrow \mathsf{RLizard.KeyGen}(params)$ is generated by sampling an instance of Ring-LWE problem with secret polynomial $s \leftarrow \mathcal{D}_s^n$. Hence, distributions $\mathcal{D}_0$ and $\mathcal{D}_1$ are computationally indistinguishable under the Ring-$\mathsf{LWE}_{n,q,\mathcal{D}G_\sigma}(\mathcal{D}_s^n)$ assumption.

Now assume that pk is uniform random over R_q^2. Then, pk and $\mathbf{c} \leftarrow \mathsf{RLizard.Enc}_{\mathsf{pk}}(0)$ together form an instance of the Ring-$\mathsf{LWR}_{n,q,p}(\mathcal{D}_r^n)$ problem so that $\mathcal{D}_1$ and $\mathcal{D}_2$ are computationally indistinguishable under the Ring-$\mathsf{LWR}_{n,q,p}(\mathcal{D}_r^n)$ assumption.

As a result, $\mathcal{D}_0$ and $\mathcal{D}_2$ are computationally indistinguishable under the Ring-$\mathsf{LWE}_{n,q,\mathcal{D}G_\sigma}(\mathcal{D}_s^n)$ and Ring-$\mathsf{LWR}_{n,q,p}(\mathcal{D}_r^n)$ assumptions. This implies the IND-CPA security of the RLizard PKE scheme. □

B. THE IND-CCA SECURE KEY ENCAPSULATION MECHANISM

We provide the full description of RLizard Key Encapsulation Mechanism, namely RLizard.KEM, in this subsection.

- $\mathsf{RLizard.KEM.Setup}(1^\lambda)$: The same as that in RLizard.Setup, additionally choosing d as a key length, hash functions $G : R_p \times R_p \times \{0,1\}^d \times R_2 \rightarrow \{0,1\}^d$, $H : R_t \rightarrow R_q$ and $H' : R_t \rightarrow \{0,1\}^d$. Output $params \leftarrow (n, t, q, p, p', d, \mathcal{D}_s, \mathcal{D}_r, \sigma, G, H, H')$.
- $\mathsf{RLizard.KEM.KeyGen}(params)$: Sample polynomials $a \leftarrow R_q$ and $s \leftarrow \mathcal{D}_s^n$. Generate a random vector $\mathbf{k} \leftarrow \{0,1\}^n$ and identity it with the polynomial $k \in R$. For $0 \leq i \leq n-1$, sample an integer $e_i \leftarrow \mathcal{D}G_\sigma$, and then set $e = \sum_{i=0}^{n-1} e_i X^i \in R_q$. Let $b = a \cdot s + e \in R_q$. Output the public key $\mathsf{pk} := (a, b) \in R_q^2$ and the secret key $\mathsf{sk} := (s, k) \in R^2$.
- $\mathsf{RLizard.KEM.Encaps}(params, \mathsf{pk})$: Generate a polynomial $\delta \leftarrow R_t$. Compute $r := H(\delta)$ and $\mathbf{d} := H'(\delta)$. Compute $c_1 := \lfloor (p/q) \cdot a \cdot r \rceil \in R_p$, $c_2 := \lfloor (p'/t) \cdot \delta \rceil + \lfloor (p'/q) \cdot b \cdot r \rceil \in R_p'$. Output $K := G(c_1, c_2, \mathbf{d}, \delta)$ and $\mathbf{c} = (c_1, c_2, \mathbf{d})$.
- $\mathsf{RLizard.KEM.Decaps}(params, \mathsf{sk}, \mathbf{c} = (c_1, c_2, \mathbf{d}))$:
  1. Parse the ciphertext $\mathbf{c} := (c_1, c_2, \mathbf{d})$.
  2. Compute $\delta' := \lfloor (t/p) \cdot ((p/p') \cdot c_2 + s \cdot c_1) \rceil \in R_t$.
  3. Compute $r' := H(\delta')$ and $\mathbf{d}' := H'(\delta')$.
  4. Compute $a' := \lfloor (p/q) \cdot a \cdot r' \rceil \in R_p$ and $b' := \lfloor (p'/t) \cdot \delta' \rceil + \lfloor (p'/q) \cdot b \cdot r' \rceil \in R_p'$ and set $\mathbf{c}' := (a', b', \mathbf{d}')$.
  5. If $\mathbf{c} \neq \mathbf{c}'$, then output $K = G(c_1, c_2, \mathbf{d}, k)$.
  6. Else, output the shared key $K = G(c_1, c_2, \mathbf{d}, \delta')$.

Since RLizard.KEM is achieved by applying a generic conversion [29] from IND-CPA secure PKE to IND-CCA secure KEM, the correctness and security results are immediate. We present formal theorems which assert our scheme achieves correctness and IND-CCA security for completeness in the following. The full proofs themselves are generic,

and neither simple nor our contributions, so we defer them to [29].

Correctness. The correctness of the RLizard PKE scheme implies that of RLizard.KEM.

Theorem 2 [29]: If the RLizard PKE scheme is correct with the probability $1-\epsilon$*, then RLizard.KEM is correct except with the probability* $1-\epsilon$ *in the (quantum) random oracle model.*

Security. The security of RLizard.KEM is derived from the theorems in [29] and the IND-CPA security of the RLizard PKE scheme. We present two theorems for the IND-CCA security of our RLizard.KEM in the random oracle model and quantum random oracle model.

Theorem 3 ([29], Theorem 2 and 3): For any IND-CCA adversary $\mathcal{B}$ *on RLizard.KEM issuing at most* q_D *queries to the decryption oracle,* q_G *queries to the random oracle* G*, and* q_H *queries to the random oracle* H*, there exists an IND-CPA adversary* $\mathcal{A}$ *on RLizard such that*

$$\mathit{Adv}^{CCA}_{\mathit{RLizard.KEM}}(\mathcal{B}) \leq q_G \cdot \epsilon + \frac{q_H}{2^{\omega(\lambda)}} + \frac{2q_G+1}{t^{\ell}} + 3 \cdot \mathit{Adv}^{CPA}_{\mathit{RLizard}}(\mathcal{A}),$$

where λ *is the security parameter and* ϵ *is the decryption failure probability of RLizard and RLizard.KEM.*

Theorem 3 follows from the fact (and Theorem 2 and 3 in [29]) that underlying PKE scheme, RLizard, is $\omega(\lambda)$-spread, which means, for every (pk, sk), possible $\mathbf{c}$ in the ciphertext space, and $m \in R_t$, $Pr_{r \leftarrow R_q}[\mathbf{c} = \mathsf{RLizard.Enc}_{\mathsf{pk}}(m; r)] \leq 2^{-\omega(\lambda)}$.

Theorem 4 ([29], Theorem 8 and 10): For any IND-CCA quantum adversary $\mathcal{B}$ *on RLizard.KEM issuing at most* q_D *(classical) queries to the decryption oracle,* q_G *queries to the quantum random oracle* G*,* q_H *queries to the quantum random oracle* H*, and* $q_{H'}$ *queries to the quantum random oracle* H'*, there exists an IND-CPA quantum adversary* $\mathcal{A}$ *on RLizard such that*

$$\mathit{Adv}^{CCA}_{\mathit{RLizard.KEM}}(\mathcal{B}) \leq (q_H + 2q_{H'})\sqrt{8\epsilon(q_G+1)^2 + (1+2q_G)\sqrt{\mathit{Adv}^{CPA}_{\mathit{RLizard}}(\mathcal{A})}},$$

where ϵ *is the decryption failure probability of RLizard and RLizard.KEM.*

We remark that Theorem 4 follows from ONE-WAY CPA security of RLizard PKE scheme (and Theorem 8 and 10 in [29]) which is a weaker notion compared to the IND-CPA security that RLizard PKE achieves.

C. PARAMETER CHOICES FOR PRACTICAL USE

We choose the parameter set for RLizard.KEM from the following perspectives:

1) The parameter sets should allow fast implementation. The most expensive operation in our RLizard.KEM key generation, encapsulation, and decapsulation is the polynomial multiplication in R_q (or R_p). Furthermore, the rounding operation for the polynomial coefficients can be expensive depending on the choices of the modulus and rounding modulus. Hence, we choose our parameters and secret distributions such that these operations are extremely simple and efficient.
2) The parameter sets should resist all computational attacks, such as dual attack [41] and primal attack [42]. We measured the attack complexities of all known (quantum) attacks for the Ring-LWE and Ring-LWR instances with respect to our parameter sets, and we concluded that the best attack complexity is greater than 2^{λ}, where λ is the security parameter.
   As far as we know, the best way to solve the Ring-LWE and Ring-LWR is to deal with them as non-structured general LWE problems. We found that attack algorithms using lattice basis reduction algorithm, especially the BKZ algorithm [43], is the most efficient in our cases. For measuring the complexity for running the BKZ algorithm in those attacks, We follow the conservative approach in NewHope paper [8] to regard the BKZ complexity as solving core-SVP by applying Grover's quantum search [44]. (We assume that SVP of a given lattice of dimension n is solved in time $2^{0.268\ n}$ according to their intensive research on quantum sieve algorithms, discarding $o(n)$ in the exponent). To sum up, time complexity of the best quantum attack for our parameter is 2^{104} which is an attack complexity for the primal attack [42], thanks to help of the online LWE estimator [45]. We remark that one can find a useful guideline for attacking the LWE problem in [46] as well. Also, we concluded that none of hybrid attacks in [47] and [48] works well, since we have errors of which distributions are close to discrete Gaussian distributions in terms of Rényi divergence (which means that errors are neither binary nor trinary) in our LWE instances.
3) Decryption failure should rarely occur for our parameters. We set our parameters such that the decryption failure rate is far lower than $2^{-\lambda}$, where λ is the security parameter.

We instantiate RLizard.KEM with a parameter set called RLizard104 for quantum 104-bit security ($\lambda = 104$). For efficiency, we choose the distribution of the secret key as $\mathcal{D}_s = \mathcal{HWT}(h_s)$ and that of the ephemeral secret in the encapsulation as $\mathcal{D}_r = \mathcal{HWT}(h_r)$. Further, we set the degree of the cyclotomic polynomial, n, and modulus $p' < p < q$ to be powers of two, and $t = 2$. In order to implement G, H, and uniform sampling, we utilize SHAKE-256 function [49] with truly random seeds.

The parameters are listed in Table 1. We also provide the estimated best attack complexity according to Albrecht's LWE estimator [45] as well as the decryption failure rate for the chosen parameter set in Table 2.

V. IMPLEMENTATION AND PERFORMANCE ANALYSIS

This section discusses the implementation aspects of RLizard. RLizard was submitted as a standard candidate for PQC standardization [16] launched by NIST in 2017.

TABLE 1. Suggested parameter set; n is the degree of the cyclotomic polynomial of the base rings, t is the size of the plaintext coefficients, q is a large modulus in Ring-LWE and Ring-LWR, p' and p are ciphertext modulus (Also, p is a rounding modulus for Ring-LWR), σ is a parameter for the discrete Gaussian distribution $\mathcal{DG}_\sigma$ in Ring-LWE, and h_s and h_r are the Hamming weights of the secret key s and ephemeral secret r, respectively.

Parameter	n	$\log q$	$\log p$	$\log p'$	σ	h_s	h_r
RLizard104	512	10	8	5	2.68	168	172

TABLE 2. Best Attack Complexity and Decryption Failure Rate for Suggested Parameter Set; T denotes the estimated best attack complexity and DFR denotes the decryption failure rate for the resulting RLizard.KEM.

Parameter	T	DFR
RLizard104	2^{104}	2^{-167}

The submission includes a reference implementation in C. However, this version of RLizard does not perform well in our reference IoT environment. In this work, we improve the performance of the RLizard implementation such that it requires much fewer clock cycles. Thus, in the improved RLizard implementation, we reduce the number of clock cycles required for key generation, encapsulation, and decapsulation by 36%, 42%, and 47%, respectively, compared to the version submitted to NIST.

We analyze the performance of the improved RLizard implementation and other KEMs in our reference environment, i.e., Atmel SAM3X8E ARM Cortex-M3 (84 Mhz, 32-bit) with 96 KB SRAM and 512 KB flash memory.

A. OVERVIEW OF THE BASIC VERSION OF IMPLEMENTATION

We provide an overview of the basic version of the RLizard implementation, which was submitted to NIST [16]. We focus on three main features that contribute significantly toward the efficiency of the operations.

1) REPRESENTING THE COEFFICIENTS OF AN ELEMENT IN R_Q (R_P)

We use q and p of the form 2^u, where u is a positive integer for making the modular reduction operation efficient. An interesting feature in this regard is that the most significant $\log_2 q$ bits in a 16-bit word-sized variable are used to represent a coefficient of an element R_q, as shown in Fig. 2-(1)-(a). Thus, we do not need to worry about the *mod* q operation when we perform some operations between coefficients. Moreover, this can be efficiently achieved through $\lfloor (q/p) \cdot c \rceil$ to embed an R_q element c into R_p by following the steps in Fig. 2-(1).

2) REPRESENTING A RING ELEMENT OF SHORT COEFFICIENTS IN R_Q

Another interesting feature of the basic implementation is that the coefficients of one ring element are "short" (i.e., one of {-1,0,1}) in every multiplication between two ring elements in RLizard. Thus, complex operations, such as number theoretic transformation, are unnecessary in the implementation.

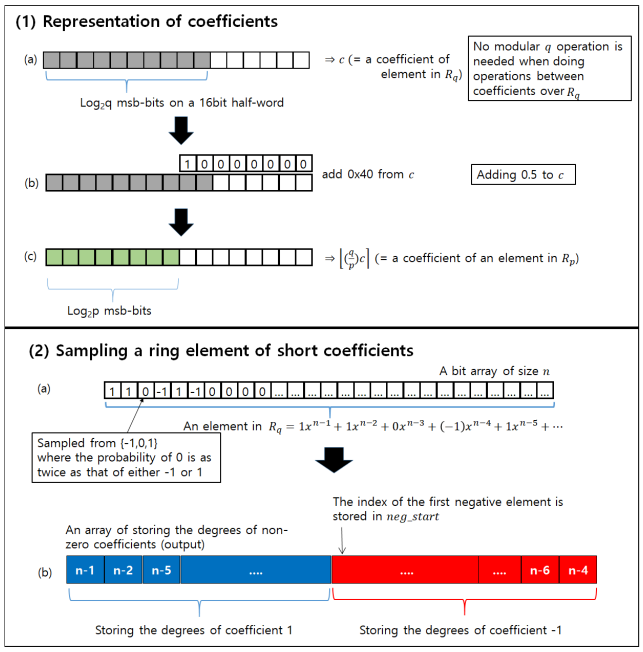

FIGURE 2. Representing the coefficients of an element in R_q (R_p) and representing a short-coefficient element.

Furthermore, in the implementation, these short-coefficient elements are represented with arrays of the degrees of non-zero coefficients. The degrees of coefficient 1 are stacked from the beginning of the array, and those of coefficient -1 are stacked backward from the end of the array. We consider the smallest index to indicate the degree of coefficient-1, which is *neg_start* in Fig. 3.

As random sampling is always performed in RLizard, for convenience, the way to represent a short-coefficient ring element is shown by randomly sampling it over the Gaussian distribution, i.e., the probability that zero is sampled is equal to half of the probability that +1 (or -1) is sampled (see Fig. 2-(2)).

If we use the representation method described above, multiplication of a short-coefficient ring element with a normal-coefficient ring element can be performed in a highly efficient manner. For example, if we multiply a ring element $a = a_{n-1} \cdot X^{n-1} + \ldots + a_1 \cdot X + a_0 \in R_q$ with another ring element shown in Fig. 2-(2)-(b), the multiplication can be expressed as $\sum_{j \in (b)} a \cdot X^j = \sum_{j \in (b)} \sum_{i=0}^{n-1} a_i X^{i+j} (mod\ X^q + 1, q)$ where (b) refers to the array shown in Fig. 2-(2)-(b).

3) EFFICIENT GAUSSIAN SAMPLING TO GENERATE NOISE

In the key generation phase, an error polynomial e should be sampled from a discrete Gaussian distribution $\mathcal{DG}_\sigma$ for a certain parameter $\sigma > 0$. We follow an efficient discrete Gaussian sampling methodology that was originally proposed in [11]. This method samples errors using the inversion sampling algorithm, which employs a precomputed look-up

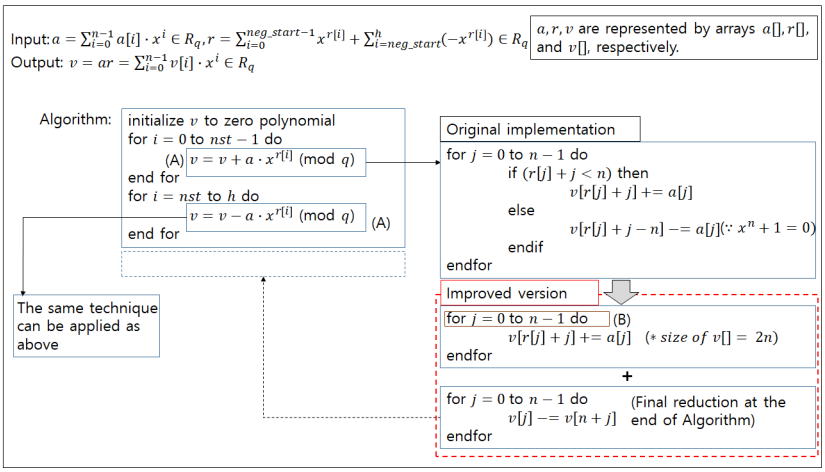

FIGURE 3. Comparison between the original implementation and the improved one: multiplication of two ring elements.

table corresponding to a discrete cumulative density function (CDF) over a small interval containing 0. The distribution of the output from this algorithm is a discrete bounded symmetric distribution that is extremely close to the discrete Gaussian distribution with respect to the Rényi divergence. Specifically, it is known that if the Rényi divergence between the desired error distribution and the target discrete Gaussian distribution is sufficiently close to 1, then exploiting the desired error distribution instead of the discrete Gaussian distribution still guarantees the security. For further details on the relation between cryptographic security and the Rényi divergence, readers may refer to [50].

B. IMPROVED IMPLEMENTATION

This subsection describes detailed methods for performance improvement over the basic version of the RLizard implementation submitted to NIST. We applied various techniques to the existing implementation in order to improve performance. Here, we focus on two methods that facilitated significant performance improvement.

According to our investigation, the operation of multiplying two R_q elements requires a significant part of the clock cycles consumed to perform the entire RLizard key encapsulation and decapsulation algorithms. One of the main advantages of RLizard in terms of efficiency is that when performing multiplication between two elements, each element's coefficients are only one of $+1$, 0, or -1. Thus, the multiplication operation can be practically implemented as modular addition between polynomial terms, which makes the operation extremely fast. However, owing to the parameter setting in consideration of safety and decryption success probability, the polynomial order n is 512 for 104-bit security and 1024 for 128-bit security; thus, the number of non-zero terms h in a multiplicand polynomial is around 170 for 104-bit security and 128 for 128-bit security. Such large parameters make the multiplication expensive.

We improved the performance of the multiplication part as follows. Fig. 3 compares the multiplication implementations of the original and improved versions. It can be seen that parts (A) and (B) are improved. Part (A) is implemented as the "Original implementation" in the NIST version. Since the polynomial reduction over $X^n + 1$ is performed simultaneously while multiplying $X^{r[j]}$ for every j, it results in many comparison instructions, as shown in the Fig. 3. By delaying the polynomial reduction until the final step of the algorithm, we could reduce the comparison instructions by hn by including n comparisons, $2n$ additions/subtractions, and $2n$ memory instructions. This improvement is applied in the "Improved version" in Fig. 3.[1] The second improvement is the reduction in the number of comparisons performed for the loop in (B). "Loop unrolling" is adopted to reduce the number of repetitions of the loop controlled by the variable j. For example, by increasing the size of the code inside the loop by 2048 bits, the number of comparisons required can be reduced from hn to $hn/64$. In addition, we achieved minor performance improvements by efficiently using randomly generated bits without wasting them, and by predicting the number of random bits used for the key generation and encapsulation operations in order to minimize the number of calls of the SHAKE-256 function [49] when generating pseudo-random bits.

Public key compression: Because every coefficient of polynomial a is sampled uniformly in $\mathbb{Z}_q$ in key generation, we can just send a 256-bit random seed instead of sending all the coefficients of a when a user wants to send his/her public key to another entity. Because we deal with the cryptosystem whose security level is below quantum 128-bit, it is enough to use 256-bit random seed to generate a random polynomial a.

[1]We have found that this idea was originally from [51] and [52] in the review process.

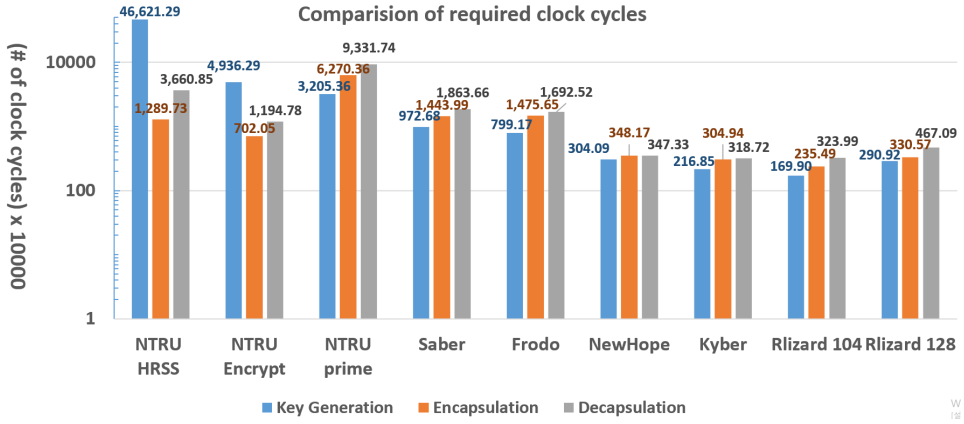

FIGURE 4. Comparison of required clock cycles; RLizard104 denotes RLizard with quantum 104-bit security, and RLizard128 denotes the parameter set in the NIST submission with quantum 128-bit security.

C. PERFORMANCE ANALYSIS IN IOT ENVIRONMENT

We analyzed the performance of RLizard and other existing KEMs in the IoT environment. Toward this end, we implemented RLizard in the IoT environment. Further, we used the source code of existing KEMs, obtained from the NIST PQC standardization web page [16], to implement these KEMs for performance comparison with RLizard. They were slightly modified for working in the IoT environment. For example, we modified them to invoke the trng_read_output_data() function when they require true random bits. We used the Arduino Due [53] to measure the performance. It has an ARM Cortex-M3 (86 Mhz) MCU with 96 KB SRAM as working memory and 512 KB flash memory to store the execution image file. We selected this environment because it effectively reflects the resource-constrained IoT environment, such as MCUs without SIMD operations and caches. Moreover, it is known that 32-bit ARM processors are the most widely used embedded processors in many high-end IoT platforms [34].

We measured the average clock cycles required for key generation, encapsulation, and decapsulation operations. We ran each scheme 10000 times and measured the average clock cycles. We also analyzed their storage requirements. We considered the maximum working memory required, the size of the executable image files, and the sizes of the public key, private key, and ciphertext.

The existing KEMs that were compared with RLizard are listed in Table 3. We focused on those KEMs that were proposed recently or presented at major security conferences. To make them run with reasonable speed in the IoT Environment, we chose their light-weight versions and corresponding security parameters, which are sufficient for IoT devices because their security level is comparable with that of AES-128 [16], which is one of the NIST standard settings that is widely used in practice. The quantum security supported by the chosen parameter settings is also stated in Table 3.

Fig. 4 shows the clock cycles required for each operation on RLizard and the schemes listed in Table 3. The x-axis of the upper graph is of log scale. We can see that RLizard with the 104-bit security setting requires the fewest clock cycles for all operations. In particular, for key encapsulation (decapsulation), RLizard with the 104-bit security setting requires 40.6% (3.79%) fewer clock cycles than the second-fastest scheme, namely *Kyber* [10].

TABLE 3. KEM schemes and their quantum security settings.

KEM Name	Parameters	Quantum security (in bits)
kyber [10]	$k = 2$	102
NewHope [8]	$n = 512$	101
Frodo [11]	$n = 640$	103
Saber [13]	Light	115
NTRU-Prime [9]	LPR	129
NTRUEncrypt [14]	$n = 443$	85
NTRU-HRSS [15]	$n = 701$	123
RLizard-104	$n = 512$	104
RLizard-128	$n = 1024$	128

Table 4 summarizes the required storage space for various KEM schemes including RLizard. Compared to the other schemes, RLizard uses a relatively large amount of storage space, except in the case of the private key size. In particular, it can be seen that the space required for the execution image file is nearly twice as large as that required in the case of other KEM schemes, such as NewHope, Frodo, and Saber. This is because the proposed scheme uses loop unrolling, which increases the code size. However, as the size of the flash memory in which these images is stored is larger than that of the SRAM in conventional IoT environments, the disadvantage due to the use of such storage space is not as severe as that due to high SRAM consumption. Further, we could not exactly measure the maximum use of SRAM in the case of Frodo [11] because it consumes all the SRAM memory when executing the test program; except for the amount of memory used by the test program, the size of the SRAM

TABLE 4. Comparison of the required storage size (in bytes); RLizard104 denotes RLizard with quantum 104-bit security, and RLizard128 denotes the parameter set in the NIST submission with quantum 128-bit security.

KEMs	Maximum SRAM use	Executable image size	private key size	public key size	ciphertext size
kyber [10]	10464	13004	1632	736	832
NewHope [8]	15120	9048	1888	928	1120
Frodo [11]	over 85879	8928	19872	9616	9736
Saber [13]	10608	9184	1568	992	736
NTRU-Prime [9]	20296	58276	1238	1047	1175
NTRUEncrypt [14]	20192	57872	701	611	611
NTRU-HRSS [15]	10160	9528	1418	1138	1278
RLizard104	12888	18748	401	672	832
RLizard128	27696	19188	385	1152	1664

TABLE 5. Power consumption ratio of other schemes compared to RLizard-104.

KEM Name	Bit length of parameters	Relative ratio of power consumption
kyber [10]	42	1.083
NewHope [8]	58	1.364
Frodo [11]	40	10.036
Saber [13]	26	2.923
NTRU-Prime [9]	34	10.622
NTRUEncrypt [14]	35	4.031
NTRU-HRSS [15]	38	27.605
RLizard-104	58	1

available is 85879 bytes. As the stack region and the heap region are overlapped while running the test program with Frodo, we could not measure the exact size of the memory required.

Power consumption analysis: The amount of power consumed is one of the most important factors to discuss the efficiency of the methods used in the IoT environment.

We compare the power consumption of the proposed method and other methods relatively. Based on the results of the research [54], we assume that the amount of power required for executing 1 clock cycle by CPU is equal to 1/1000 of the power for 1 bit data transmission.

Under this assumption, we compare power consumption of the proposed method with those of other methods. The data transmitted in one KEM execution are a public key and its related parameters, and a ciphertext. In addition, since the key generation algorithm is performed only once in the entire life time of devices, it is assumed that only the key encapsulation and the key decapsulation algorithm are performed when a KEM is executed once. Based on this assumption, we calculate the number of clock cycles required for one KEM execution. The bit length of the parameters are estimated by analyzing the shortest bit lengths with which each parameter can be expressed in each method.

Considering all of them, we provide the results of analyzing the required amount of power for other methods when the amount of power required to perform the KEM once using the proposed method is 1, in the following Table 5. From the result of the analysis of the power amount, we can say that the power consumption of the proposed method is the lowest.

TABLE 6. Ratio of the required number of clock cycles of RLizard-104 algorithms in ARC embedded processor (32MHz) environment compared to those in ARM COrtex-M3 (84Mhz) environment.

Key Generation	Encapsulation	Decapsulation
97.47%	94.13%	96.60%

Performance in a different CPU environment: we implemented RLizard-104 in the ARC embedded processor (32MHz), 24KB SRAM, 192KB Flash memory environment to verify the performance of RLizard-104. The result of measuring the performance of each algorithm is shown in the following Table 6. The result shows that the number of clock cycles used is 2.3% $\sim$ 5.8% less than in the Cortex-M3 (84MHz) environment.

VI. CONCLUSION

In this paper, we proposed RLizard as a pq-KEM that can be used in the IoT environment. The security of the proposed KEM depends on the RLWE/RLWR problems; hence, the key generation time and the space required for key management are reduced compared to the existing Lizard scheme. Further, we proposed an improved implementation compared to the version submitted for pq-KEM standardization. As a result of the improvement, we confirmed that, in the parameter setting with AES-128-level security, RLizard consumes the fewest clock cycles in the 32-bit ARM environment compared to other KEM schemes that have been submitted for pq-KEM standardization. Our experiments showed that RLizard is well suited for the IoT environment, which has strict requirements in terms of computational resources but relatively relaxed requirements in terms of the supported storage space. In addition, the performance analysis of RLizard and other pq-KEMs in the same environment is expected to provide important reference data for future improvement of the implementation performance of pq-KEMs.

ACKNOWLEDGMENT

(Joohee Lee (skfro6360@snu.ac.kr), Duhyeong Kim (doodoo1204@snu.ac.kr), and Hyungkyu Lee (hklee@nslab.kaist.ac.kr) are co-first authors.) The authors would like to thank Seungwan Hong for supportive assistance with a python code for evaluating decryption failure rates.

REFERENCES

[1] Statistica. (2018). *Internet of Things (IoT) Connected Devices Installed Base Worldwide From 2015 to 2025 (in Billions)*. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] R. Minerva, A. Biru, and D. Rotondi, "Towards a definition of the Internet of Things (IoT)," *IEEE Internet Initiative*, to be published.

[3] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of Things: Security vulnerabilities and challenges," in *Proc. 3rd IEEE Int. Workshop Smart City Ubiquitous Comput. Appl.*, Jul. 2015, pp. 180–187.

[4] Z. Ling, J. Luo, Y. Xu, C. Gao, K. Wu, and X. Fu, "Security vulnerabilities of Internet of Things: A case study of the smart plug system," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1899–1909, Dec. 2017.

[5] P. Pongle and G. Chavan, "A survey: Attacks on RPL and 6LowPAN in IoT," in *Proc. Int. Conf. Pervasive Comput.*, Jan. 2015, pp. 1–6, doi: 10.1109/PERVASIVE.2015.7087034.

[6] A. Sajid, H. Abbas, and K. Saleem, "Cloud-assisted IoT-based SCADA systems security: A review of the state of the art and future challenges," *IEEE Access*, vol. 4, pp. 1375–1384, 2016.

[7] C. Cheng, R. Lu, A. Petzoldt, and T. Takagi, "Securing the Internet of Things in a quantum world," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 116–120, Feb. 2017.

[8] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange—A new hope," in *Proc. USENIX Secur. Symp.*, 2016, pp. 327–343.

[9] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and C. van Vredendaal, "NTRU prime: Reducing attack surface at low cost," in *Proc. Int. Conf. Sel. Areas Cryptogr.*, in Lecture Notes in Computer Science, vol. 10719, 2017, pp. 235–260.

[10] J. Bos *et al.*, "CRYSTALS—Kyber: A CCA-secure module-lattice-based KEM," in *Proc. 3rd IEEE Eur. Symp. Secur. Privacy*, Apr. 2018, pp. 353–367.

[11] J. Bos *et al.*, "Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1006–1018.

[12] 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 *Proc. 11th Conf. Secur. Cryptogr. Netw.*, Sep. 2018, pp. 160–177.

[13] J.-P. D'Anvers, A. Karmakar, S. S. Roy, F. Vercauteren, "Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM," in *Proc. Int. Conf. Cryptol. Africa*, in Lecture Notes in Computer Science, vol. 10831, 2018, pp. 282–305.

[14] J. Hoffstein *et al.*, "Choosing parameters for NTRUEncrypt," in *Proc. Topics Cryptol.-CT-RSA*, in Lecture Notes in Computer Science, vol. 10159, 2017, pp. 3–18.

[15] A. Hülsing, J. Rijneveld, J. Schanck, and P. Schwabe, "High-speed key encapsulation from NTRU," in *Proc. Int. Conf. Cryptograph. Hardw. Embedded Syst.*, in Lecture Notes in Computer Science, vol. 10529, 2017, pp. 232–252.

[16] National Institute of Standards and Technology. (2017). *Post-Quantum Cryptography Standardization*. [Online]. Available: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Post-Quantum-Cryptography-Standardization

[17] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in *Proc. Adv. Cryptol.-Eurocrypt*, in Lecture Notes in Computer Science, vol. 6110, 2010, pp. 1–23.

[18] D. Stehlé, R. Steinfeld, K. Tanaka, and K. Xagawa, "Efficient public key encryption based on ideal lattices," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2009, pp. 617–635.

[19] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proc. 37th Annu. ACM Symp. Theory Comput. (STOC)*, 2005, pp. 84–93.

[20] C. Peikert, O. Regev, and N. Stephens-Davidowitz, "Pseudorandomness of Ring-LWE for Any Ring and Modulus," in *Proc. 49th Annu. ACM SIGACT Symp. Theory Comput.*, 2017, pp. 461–473.

[21] M. Chase *et al.* (2017). *Security of Homomorphic Encryption, Draft Homomorphic Encryption Standard*. [Online]. Available: HomomorphicEncryption.org

[22] W. Castryck, I. Iliashenko, and F. Vercauteren, "Provably weak instances of Ring-LWE revisited," in *Proc. Adv. Cryptol.-EUROCRYPT*, in Lecture Notes in Computer Science, vol. 9665, 2016, pp. 147–167.

[23] H. Chen, K. E. Lauter, and K. E. Stange, "Vulnerable Galois RLWE families and improved attacks," in *Proc. IACR Cryptol. ePrint Arch.*, 2016, pp. 1–15.

[24] Y. Elias, K. E. Lauter, E. Ozman, and K. E. Stange, "Provably weak instances of Ring-LWE," in *Proc. Annu. Cryptol. Conf. (Crypto)*, 2015, pp. 63–92.

[25] A. Banerjee, C. Peikert, and A. Rosen, "Pseudorandom functions and lattices," in *Proc. Annu. Int. Conf. Theory Appl. Cryptograph. Techn.*, 2012, pp. 719–737.

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

[27] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Proc. Int. Algorithmic Number Theory Symp.*, 1998, pp. 267–288.

[28] N. Howgrave-Graham, J. H. Silverman, A. Singer, W. Whyte, and N. Cryptosystems, "NAEP: Provable security in the presence of decryption failures," *IACR Cryptol. ePrint Arch.*, 2003, p. 172.

[29] D. Hofheinz, K. Hövelmanns, E. Kiltz, and Y. Kalai, "A modular analysis of the Fujisaki–Okamoto transformation," in *Proc. Theory Cryptogr.*, 2017, pp. 341–371.

[30] R. Lindner and C. Peikert, "Better key sizes (and attacks) for LWE-based encryption," in *Topics in Cryptology—CT-RSA* (Lecture Notes in Computer Science), vol. 6558. Berlin, Germany: Springer, 2011, pp. 319–339.

[31] O. M. Guillen, T. Pöppelmann, J. M. B. Mera, E. F. Bongenaar, G. Sigl, and J. Sepulveda, "Towards post-quantum security for IoT endpoints with NTRU," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, 2017, pp. 698–703.

[32] A. Boorghany, S. B. Sarmadi, and R. Jalili, "On constrained implementation of lattice-based cryptographic primitives and schemes on smart cards," *ACM Trans. Embed. Comput. Syst.*, vol. 14, no. 3, Apr. 2015, Art. no. 42.

[33] Z. Liu *et al.*, "Efficient ring-LWE encryption on 8-bit AVR processors," in *Proc. Cryptograph. Hardw. Embedded Syst. (CHES)*, in Notes in Computer Science, vol. 9293, 2015, pp. 663–682.

[34] Z. Liu, R. Azarderakhsh, H. Kim, and H. Seo, "Efficient implementation of ring-LWE encryption on high-end IoT platform," in *Proc. RFIDSec*, in Notes in Computer Science, vol. 10155, 2017, pp. 76–90.

[35] Z. Liu, R. Azarderakhsh, H. Kim, and H. Seo, "Efficient software implementation of ring-LWE encryption on IoT processors," *IEEE Trans. Comput.*, to be published, doi: 10.1109/TC.2017.2750146.

[36] S.-H. Seo, J. Won, and E. Bertino, "pCLSC-TKEM: A pairing-free certificateless signcryption-tag key encapsulation mechanism for a privacy-preserving IoT," *Trans. Data Privacy*, vol. 9, no. 2, pp. 101–130, 2016.

[37] W. Wang, P. Xu, and L. T. Yang, "One-pass anonymous key distribution in batch for secure real-time mobile services," in *Proc. IEEE Int. Conf. Mobile Services*, Jun./Jul. 2015, pp. 158–165.

[38] J. Cheon *et al.* (Oct. 2018). *Algorithm Specification of Lizard, Post-Quantum Cryptography Standardization 1st Round submission*. [Online]. Available: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions/Lizard.zip

[39] C. Gentry, C. Peikert, and V. Vaikuntanathan, "Trapdoors for hard lattices and new cryptographic constructions," in *Proc. 14th Annu. ACM Symp. Theory Comput. (STOC)*, 2008, pp. 197–206.

[40] D. Micciancio and C. Peikert, "Trapdoors for lattices: Simpler, tighter, faster, smaller," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 7237. Berlin, Germany: Springer-Verlag, 2012, pp. 700–718.

[41] M. R. Albrecht, "On dual lattice attacks against small-secret LWE and parameter choices in HElib and SEAL," in *Proc. Adv. Cryptol.-Eurocrypt*, in Lecture Notes in Computer Science, vol. 10211, 2017, pp. 103–129.

[42] M. R. Albrecht, F. Göpfert, F. Virdia, and T. Wunderer, "Revisiting the expected cost of solving uSVP and applications to LWE," in *Proc. Adv. Cryptol.-Asiacrypt*, in Lecture Notes in Computer Science, vol. 10624, 2017, pp. 297–322.

[43] Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," in *Proc. Adv. Cryptol.-Asiacrypt*, in Lecture Notes in Computer Science, vol. 7073, 2011, pp. 1–20.

[44] T. Laarhoven, M. Mosca, and J. van de Pol, "Finding shortest lattice vectors faster using quantum search," *Des., Codes Cryptogr.*, vol. 77, nos. 2–3, pp. 375–400, 2015.
[45] M. R. Albrecht. (2017). *A Sage Module for Estimating the Concrete Security of Learning With Errors Instances*. [Online]. Available: https://bitbucket.org/malb/lwe-estimator
[46] M. R. Albrecht, R. Player, and S. Scott, "On the concrete hardness of learning with errors," *J. Math. Cryptol.*, vol. 9, no. 3, pp. 169–203, 2015.
[47] F. Göpfert, C. van Vredendaal, and T. Wunderer, "A hybrid lattice basis reduction and quantum search attack on LWE," in *Proc. Int. Workshop Post-Quantum Cryptogr.*, 2017, pp. 184–202.
[48] N. Howgrave-Graham, "A hybrid lattice-reduction and meet-in-the-middle attack against NTRU," in *Proc. Annu. Cryptol. Conf. (Crypto)*, 2007, pp. 150–169.
[49] M. J. Dworkin, "SHA-3 standard: Permutation-based hash and extendable-output functions," NIST, Gaithersburg, MA, USA, Tech. Rep. 202, 2015.
[50] S. Bai, T. Lepoint, A. Roux-Langlois, A. Sakzad, D. Stehlé, and R. Steinfeld, "Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance," *J. Cryptol.*, vol. 31, no. 2, pp. 610–640, 2018.
[51] S. Akleylek, E. Alkım, and Z. Y. Tok, "Sparse polynomial multiplication for lattice-based cryptography with small complexity," *J. Supercomput.*, vol. 72, no. 2, pp. 438–450, 2016.
[52] S. Akleylek, N. Bindel, J. Buchmann, J. Krämer, and G. A. Marson, "An efficient lattice-based signature scheme with provably secure instantiation," in *Proc. AFRICACRYPT*, in Lecture Notes in Computer Science, vol. 9646, 2016, pp. 44–60.
[53] S. Monk, *Programming Arduino Next Steps: Going Further With Sketches*. New York, NY, USA: McGraw-Hill, 2014.
[54] V. Raghunathan, C. Schurgers, S. Park, and M. B. Srivastava, "Energy-aware wireless microsensor networks," *IEEE Signal Process. Mag.*, vol. 19, no. 2, pp. 40–50, Mar. 2002.
[55] C. Peikert, "Lattice cryptography for the Internet," in *Proc. Int. Workshop Post-Quantum Cryptogr.*, 2014, pp. 197–219.

Authors' photographs and biographies not available at the time of publication.