





Article

NTRU-Like Random Congruential Public-Key Cryptosystem for Wireless Sensor Networks [†]

Anas Ibrahim ^{1,2,*} , Alexander Chefranov ², Nagham Hamad ³ , Yousef-Awwad Daraghmi ¹ , Ahmad Al-Khasawneh ^{4,5} and Joel J. P. C. Rodrigues ^{6,7} 

¹ Department of Computer Systems Engineering, Palestine Technical University—Kadoorie, Tulkarm 7, Palestine; y.awwad@ptuk.edu.ps

² Department of Computer Engineering, Eastern Mediterranean University, Famagusta 99628, North Cyprus via Mersin 10, Turkey; alexander.chefranov@emu.edu.tr

³ Department of Information Technology, Palestine Technical University—Kadoorie, Tulkarm 7, Palestine; nagham.hamad@ptuk.edu.ps

⁴ Irbid National University, P.O. Box 2600, Irbid 21110, Jordan; akhasawneh@hu.edu.jo

⁵ Department of Computer Information System, Hashemite University, P.O. Box 150459, Zarqa 13115, Jordan

⁶ Federal University of Piauí, Teresina 64049-550, PI, Brazil; joeljr@ieee.org

⁷ Instituto de Telecomunicações, 6201-001 Covilhã, Portugal

* Correspondence: a.melhem@ptuk.edu.ps

[†] This paper is an extended version of our paper published in Ibrahim, A.; Chefranov, A.; Hamad, N.

NTRU-Like Secure and Effective Congruential Public-Key Cryptosystem Using Big Numbers.

In Proceedings of the 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), Amman, Jordan, 9–11 October 2019; pp. 20–26.

Received: 17 April 2020; Accepted: 10 August 2020; Published: 17 August 2020



Abstract: Wireless sensor networks (WSNs) are the core of the Internet of Things and require cryptographic protection. Cryptographic methods for WSN should be fast and consume low power as these networks rely on battery-powered devices and microcontrollers. NTRU, the fastest and secure public key cryptosystem, uses high degree, N , polynomials and is susceptible to the lattice basis reduction attack (LBRA). Congruential public key cryptosystem (CPKC), proposed by the NTRU authors, works on integers modulo q and is easily attackable by LBRA since it uses small numbers for the sake of the correct decryption. Herein, RCPKC, a random congruential public key cryptosystem working on degree $N = 0$ polynomials modulo q , is proposed, such that the norm of a two-dimensional vector formed by its private key is greater than \sqrt{q} . RCPKC works as NTRU, and it is a secure version of insecure CPKC. RCPKC specifies a range from which the random numbers shall be selected, and it provides correct decryption for valid users and incorrect decryption for an attacker using LBRA by Gaussian lattice reduction. RCPKC asymmetric encryption padding (RAEP), similar to its NTRU analog, NAEP, is IND-CCA2 secure. Due to the use of big numbers instead of high degree polynomials, RCPKC is about 27 times faster in encryption and decryption than NTRU. Furthermore, RCPKC is more than three times faster than the most effective known NTRU variant, BQTRU. Compared to NTRU, RCPKC reduces energy consumption at least thirty times, which allows increasing the life-time of unattended WSNs more than thirty times.

Keywords: wireless sensor network; random congruential public-key cryptosystem; lattice; NTRU; polynomial; lattice basis reduction attack; LLL algorithm; Gaussian lattice reduction; IND-CCA2 security

1. Introduction

Wireless sensor networks (WSNs) play an important role in the development of the Internet of Things (IoT). WSNs consist of a large number of sensor nodes, battery-supplied devices with a limited memory and computational power microcontroller. WSNs are used widely, e.g., in environmental practices, health, industrial control, military [1], multimedia networks [2], and smart grid networks [3]. WSNs need security and confidentiality since sensitive information is stored, processed, or transferred by sensor nodes [4]. Therefore, cryptographic schemes efficiently working on limited WSN microcontrollers are demanded [5]. Furthermore, energy savings is very important for WSNs [6]. NTRU [7,8] is a public key cryptosystem (PKC) standardized as IEEE P1363.1 and is faster than RSA and ECC [9], and it is applicable to WSNs [10]. Contrary to RSA and ECC working with big numbers and homomorphic only in one operation, multiplication and addition, respectively, NTRU works with high degree, N , polynomial rings and is homomorphic with respect to both multiplication and addition [11]. These features of NTRU allow its use in various applications, such as authentication for smart cards [12], encryption of user data in smart monitoring systems [13], securing of SMS [14], mutual authentication and key agreement for wireless communications [15], embedded systems including microcontrollers and FPGAs [16], Internet of Things devices [17], and NTRU hardware implementation [18]. The NTRU model expects that the public key is used for encryption only by a public user (sender), whereas the private key is used for decryption by the key's owner (receiver).

NTRU and its known variants [7,8,19–38], shown in Section 2, work with degree N polynomials. The main problem NTRU faces is that it is susceptible to the lattice basis reduction attack (LBRA) using the Gaussian lattice reduction (GLR) algorithm for two-dimensional lattices and the LLL algorithm for higher dimensions [39]. The LBRA using LLL algorithm solves the shortest vector problem (SVP) with exponential in N running time revealing the secret key because the private keys are selected as polynomials with small coefficients for the decryption correctness [40]. To overcome the problem of susceptibility, NTRU uses large N resulting in high computational complexity [11,41]. Therefore, NTRU variants, shown in Section 2, try minimizing NTRU computational complexity by extending the coefficients of the polynomials used or using matrices of polynomials that allow preserving the security level while decreasing the polynomial degree. The extreme case is a polynomial of zero degree, that is integers modulo $q \gg 1$, as used in the congruential public key cryptosystem (CPKC), but CPKC with the NTRU encryption/decryption mechanism is insecure against LBRA by GLR (crackable in about 10 iterations) ([26], pp. 373–376, 451). Therefore, the CPKC is considered as a toy model of NTRU because “it provides the lowest dimensional introduction to the NTRU public key cryptosystem” [26] (p. 374). The insecurity of CPKC stems from the choice of the private keys used as small numbers to provide decryption correctness. If CPKC could be made resistant to GLR attack, it would be the best possible choice for the NTRU modifications. Therefore, we propose a CPKC modification, random CPKC (RCPKC) [42] (we call it here RCPKC.1).

In this paper, an enhanced RCPKC is proposed by specifying a range from which the random numbers shall be selected based on short vectors returned by GLR attack on it. It provides correct decryption for valid users and incorrect decryption for an attacker using GLR. GLR cannot find its private key because it solves SVP returning the shortest in a lattice vector, whereas our private key is in the safe region (above Minkowski's boundary (27)–(30) for the shortest vector norm of a lattice). On the other hand, the short vectors returned by GLR cannot be used for correct decryption due to our choice of the random numbers. RCPKC is a cryptosystem more secure than NTRU because LBRA is currently considered as one of the most effective attacks against NTRU, and also, a number of other attacks on NTRU are not applicable to RCPKC, whereas RCPKC's resistance to other known attacks on NTRU is similar to that of NTRU. RCPKC is about 27 times faster in encryption and decryption than NTRU. Simplicity, speed, and security make RCPKC a good candidate cryptosystem for WSNs. The paper's contribution can be summarized as follows:

- RCPKC, an NTRU-like cipher variant resistant to lattice based attacks, is proposed with enhanced security compared to RCPKC.1 [42].
- The hardness of the RCPKC one-way (OW) function is proven.
- RCPKC symmetric encryption padding (RAEP) is IND-CCA2 secure is proven under the assumption of the hardness of inverting an associated one-way function
- RCPKC's performance is justified through implementation and comparison with the state-of-the-art ciphers.
- RCPKC's better applicability than NTRU to WSNs is proven.

The rest of the paper is organized as follows. In Section 2, known NTRU variants are presented. In Section 3, an overview of NTRU, NTRU AEP (NAEP), the IND-CCA2 security of NAEP, and CPKC is given, and formulas for CPU power consumption calculation are introduced. LBRA by GLR on CPKC is described, and Minkowski's second theorem is presented in Section 4, used to define a region where GLR attack against the CPKC private key/message fails. In Section 5, RCPKC is presented. In Section 6, the RCPKC security comparison versus NTRU is conducted. In Section 7, the RCPKC OW function and RCPKC asymmetric encryption padding (RAEP) IND-CCA2 security are considered. In Section 8, the RCPKC performance comparison versus NTRU and its variants is presented, and the RCPKC versus NTRU power consumption is studied. Section 9 concludes the paper.

2. Review of Known NTRU Variants

Many variants of NTRU have been proposed and studied recently, targeting further decreasing its computational complexity. All these variants work with polynomials and mainly differ in the choice of their coefficients, the ring defining polynomial, or the polynomials used as the entries of such structures as matrices. The NTRU variants' overview follows.

NTRU variants differing in the choice of their coefficients. In [27], an NTRU variant, ETRU, was proposed working with polynomials over Eisenstein integer coefficients and was faster than NTRU in encryption/decryption by 1.45/1.72 times. Karbasi and Atani [28] modified ETRU, called ILTRU [28], so that it works with irreducible cyclotomic polynomial over Eisenstein integer coefficients. An NTRU variant, BITRU, working with polynomials over so-called binary numbers, usually known as complex numbers, was proposed in [20]. An NTRU variant, QTRU, working with polynomials over hyper-complex four-component numbers, quaternions, was proposed in [30]. Furthermore, Bagheri and colleagues proposed an NTRU variant, BQTRU, working over quaternions, but with bivariate polynomials, seven times faster than NTRU in encryption [21]. A variant of NTRU working with polynomials over eight-component hyper-complex numbers, octonions, called OTRU, was proposed in [29]. In [34], an NTRU variant, HXDTRU, was proposed working with polynomials over 16-component hyper-complex numbers, hexadecnions, also known as sedenions [19]. Furthermore, a variant of NTRU working with polynomials over 16-component hyper-complex numbers, sedenions, was proposed in [31]. A variant of NTRU, called CTRU, working with polynomials, the coefficients of which are also polynomials in one variable over a binary field, was proposed in [24]. Furthermore, a variant of NTRU working with polynomials, the coefficients of which are polynomials in one variable over a rational field, called BTRU, was proposed in [32].

NTRU variants working with different rings. An NTRU variant that works with polynomials with prime cyclotomic rings was proposed in [35]. A variant of NTRU working with non-invertible polynomials was proposed in [22].

NTRU variants working with polynomials inside more complicated structures. An NTRU variant working with square matrices of polynomials was proposed in [23] and was shown to be 2.5 times better than NTRU encryption and decryption time. An NTRU variant, called NNRU, working with polynomials also being entries of square matrices forming a non-commutative ring, was proposed in [33]. Apart from the polynomial variants, an NTRU-like cipher over the ring of integers, called ITRU, was proposed in [25] without referencing CPKC [26]. In ITRU [25], Table 1, p. 34, the NTRU model specified above was given, but a model for the proposed ITRU was not defined. Its Algorithm 1, [25],

p. 35, describes the key generation, and hence, it shall be made by the key owner (receiver). On the other hand, in Section 4. A, Parameter selection, [25], p. 37, the most important parameter, q , was selected by the sender (which encrypts a message using the public key, $h' = 423,642$, and random value, $r' = 19$, in [25], (19), p. 37 with the help of the private keys, f', g' , which contradicts the NTRU model: the secret key is known to only the key owner that uses the private key only for decryption, whereas the public key is used for encryption by the public user only.

Thus, the NTRU variants try minimizing NTRU's computational complexity by extending coefficients of the polynomials used or using matrices of polynomials that allow preserving the security level while decreasing the polynomial degree because operations with high-degree polynomials are time-consuming. However, these variants are still susceptible to LBRA using LLL, but with less complexity than NTRU has. Furthermore, ITRU can be used by the key owner only for encryption and decryption messages, but cannot be used by a public user knowing the public key only; hence, it is not compatible with the NTRU model of use.

3. Preliminaries

3.1. Overview of NTRU

NTRU [7,8] uses the rings:

$$R_q = \frac{\mathbb{Z}_q[x]}{x^N - 1}, R_p = \frac{\mathbb{Z}_p[x]}{x^N - 1},$$

elements of which are polynomials modulo $x^N - 1$ with coefficients in $\mathbb{Z}_q, \mathbb{Z}_p$, respectively, where $p = 2$, $q > 4d + 1$ is prime, $N > 3k + 8$ is prime, k is the security parameter, and d is the minimal integer such that $\binom{N/2}{d/2} / \text{sqrt}(N) > 2^k$, where $\binom{m}{n}$ is the number of combinations of n elements out of m .

Secret polynomials, f and g , are binary polynomials from D_f and D_g , with $d_f = d_g = d$ coefficients equal to one. Both f and g are invertible modulo q . Public key, h , is defined as:

$$h = p \cdot F_q \cdot g \text{ mod } q, \quad (1)$$

where F_q is the inverse of f modulo q . A binary message, $m = D_m \in R_p$, is encrypted using a random binary polynomial r from D_r with the $d_r = \lfloor N/2 \rfloor$ ones as follows:

$$e = F_{h,p}^{NTRU}(m, r) = p \cdot r \cdot h + m \text{ mod } q. \quad (2)$$

NTRU decryption consists of two steps:

Step 1: The private key, f , is applied to (2):

$$\begin{aligned} a &= f \cdot e \text{ mod } q \\ &= p \cdot r \cdot g + f \cdot m. \end{aligned} \quad (3)$$

Step 2: The inverse of f modulo p is applied to (3) after the polynomial a is centered using the center() function. An implementation of center(), called center1(), provided in [43] (p. 4), follows:

1. Calculate $m(1)$ as $e(1) - p \cdot r(1) \cdot h(1)$, reduced to the interval,

$$\frac{N - q}{2} \leq m(1) < \frac{N + q}{2}. \quad (4)$$

2. Denote a reduced to the interval $[0, q - 1]$ by \underline{a} . The underline is intended to indicate the minimal possible interval.
3. Calculate $a(1)$. This will differ from $p \cdot r(1) \cdot h(1) + f(1) \cdot m(1)$ by $k \cdot q$, for some integer, k .
4. Add q to the lowest k entries of a to obtain a reduced into the correct interval.

NTRU decryption fails if the following condition does not hold,

$$\text{Width}(p \cdot g \cdot r + f \cdot m) < q. \quad (5)$$

where $\text{Width}(p(x)) = \max_{i=0, \dots, N-1} (p_i) - \min_{i=0, \dots, N-1} (p_i)$. Application of the center() function to the left-hand side (LHS) of (3) makes the second equality in (3) true under Condition (5). For the conditions imposed on NTRU parameters described above, in particular, $p = 2, q > 4d + 1$, the second equality in (3) holds, and there is no need for centering.

3.2. NTRU Asymmetric Encryption Padding IND-CCA2 Security

NTRU asymmetric encryption padding (NAEP) has been proven IND-CCA2 secure [7]. In Section 3.2.1, NAEP is introduced, and in Section 3.2.2, its IND-CCA2 security is discussed.

3.2.1. NAEP Description

NAEP uses a function,

$$\text{compress}(p(x)) = p(x) \bmod q \bmod 2, \quad (6)$$

where $p(x)$ is a polynomial. NAEP encryption is introduced in Algorithm 1.

Algorithm 1: NAEP encryption.

input : $N = \theta(k); N > l = \theta(k)$ is the padding size; $G : \{0, 1\}^{N-l} \times \{0, 1\}^l \rightarrow D_r$ and $H : \{0, 1\}^N \rightarrow \{0, 1\}^N$ are hash functions; $m \in \{0, 1\}^{N-l}$ is the input plaintext message; h is the public key; q is the modulus value.

output: $e \in R_q$ is the ciphertext.

begin

1. Pick $\mu \leftarrow_R \{0, 1\}^l$, where \leftarrow_R means uniform random sampling
2. Let $\rho = G(m, \mu), r = \text{genr}(\rho), s = \text{compress}(p \cdot r \cdot h)$, and $\omega = (m || \mu) \oplus H(s)$ // genr is // a function generating correct r ; \oplus denotes XOR;
3. If $\omega \notin \tilde{R}$, goto 1 // \tilde{R} is the space of binary polynomials with the number of ones such // that the probability of NTRU decryption failure is negligible.
4. $e = F_{h,p}^{\text{NTRU}}(\omega, r)$ // according to (2)

end

The compress() binary string result is used in Step 2 of NAEP encryption to hide the padded message by the XOR operation and hashing. NAEP decryption is introduced in Algorithm 2.

Algorithm 2: NAEP decryption.

input : $N = \theta(k)$; $N > l = \theta(k)$ is the padding size; $G : \{0, 1\}^{N-l} \times \{0, 1\}^l \rightarrow D_r$ and $H : \{0, 1\}^N \rightarrow \{0, 1\}^N$ are hash functions; $e \in R_q$ is the ciphertext; Output: $m \in \{0, 1\}^{N-l}$ is the decrypted plaintext message if decrypted correctly, and Reject otherwise.

output: $m \in \{0, 1\}^{N-l}$ is the decrypted plaintext message if decrypted correctly, and Reject otherwise.

begin

1. $a = \text{center}(f \cdot e \bmod q)$
2. $\omega = F_p \cdot a \bmod p$ // according to NTRU Step 2
3. $s = \text{compress}(e - \omega)$
4. $m || \mu = \omega \oplus H(s); r = \text{genr}(G(m || \mu))$.
5. if $p \cdot r \cdot h = s \bmod q$, then output m ; else, output Reject.

end

3.2.2. NAEP IND-CCA2 Security

NAEP has been proven to be IND-CCA2 secure [7].

Definition 1. [7] (p. 3): A time τ algorithm \mathcal{A} is a $(\tau; \epsilon)$ -chosen ciphertext algorithm, with advantage ϵ in attacking a randomized encryption scheme $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ if there is a pair of sub-algorithms

$$\mathcal{A}_1 : PK \rightarrow M \times M \times S,$$

$$\mathcal{A}_2 : C \times S \rightarrow \{0, 1\},$$

such that if $(m_0, m_1, s) \leftarrow \mathcal{A}_1(pk)$, then:

$$\text{Prob}(\mathcal{A}_2(c^*, s) = b) - \frac{1}{2} = \frac{1}{2}\epsilon,$$

where $c^* \leftarrow \mathcal{E}(m^*, r^*)$, for some $r^* \in R_{\mathcal{E}}$ and $m^* = m_{b^*}$ for some $b^* \in \{0, 1\}$. This probability is defined over the choice of $r \leftarrow_R R_{\mathcal{E}}$, $b^* \in \{0, 1\}$, and $k \in R_{\mathcal{K}}$, where $R_{\mathcal{E}}$ and $R_{\mathcal{K}}$ are defined below.

The algorithms $(\mathcal{A}_1, \mathcal{A}_2)$ have access to a decryption oracle \mathcal{D} , which they can call on all but the challenge ciphertext c^* , but they must make all hash function calls to H_1, \dots, H_n public.

An encryption scheme is IND-CCA2 secure if there exist no polynomial (on security parameter) time adversary with a non-negligible advantage. Key generation, encryption, and decryption algorithms are formalized as follows [7]. For a given parameter set P , the encryption scheme is specified by three algorithms:

$$\mathcal{K} : R_{\mathcal{K}} \rightarrow PK \times SK,$$

$$\mathcal{E} : PK \times M \times R_{\mathcal{E}} \rightarrow C,$$

$$\mathcal{D} : SK \times C \rightarrow M,$$

called the key generation, encryption, and decryption algorithms, respectively. The spaces $R_{\mathcal{K}}, PK, SK, M, R_{\mathcal{E}}, C$ are called the key-gen randomness, public key, secret key, message, encryption randomness, and ciphertext space, respectively.

If $(pk_k, sk_k) \leftarrow \mathcal{K}(k)$, then the algorithms should satisfy:

$$\mathcal{D}(sk_k, \mathcal{E}(pk_k, m, r)) = m$$

for all $k \in R_K$, $m \in M$ and $r \in R_E$.

NTRU key, encryption, and decryption procedures and respective spaces are defined in Section 3.1 according to [7]. Polynomials used in NAEP [7,8] for keys are invertible. The NTRU one-way problem is defined as follows:

Definition 2. *NTRU-OW problem:* For a parameter set, \mathcal{P}_{NTRU} , we denote by $Succ_{NTRU}^{OW}(\mathcal{A}, \mathcal{P}_{NTRU})$ the success probability of a probabilistic polynomial time (PPT) adversary, \mathcal{A} , for finding a pre-image of $F_{h,p}^{NTRU}$,

$$Succ_{NTRU}^{OW}(\mathcal{A}, \mathcal{P}_{NTRU}) = \Pr \left(\begin{array}{l} (m', r') \leftarrow \mathcal{A}(e, h) \\ \text{such that } F_h^{NTRU}(m', r') = e \end{array} \right).$$

Assumption 1. *NTRU-OW assumption:* For every PPT adversary, \mathcal{A} , solving the NTRU-OW problem, there exists a negligible function, $v_A(k)$, such that for sufficiently large k , it holds:

$$Succ_{NTRU}^{OW}(\mathcal{A}, \mathcal{P}_{NTRU}) \leq v_A(k).$$

NTRU variants [7,40] can fail (see Section 6.5); hence, it was assumed in [7] that the failure probability is negligible. Under these assumptions, the IND-CCA2 security of NAEP was proven in [7], Corollary 1.

3.3. Overview of CPKC

Two secret integers, f and g , are defined as follows:

$$f < \sqrt{q/2}, \quad \sqrt{q/4} < g < \sqrt{q/2}, \quad (7)$$

$$\gcd(f, qg) = 1, \quad (8)$$

where q is a public integer.

The first secret value, f , has inverses modulo g and q , that is F_g and F_q , respectively, by virtue of (8):

$$1 = f \cdot F_g \bmod g, \quad 1 = f \cdot F_q \bmod q. \quad (9)$$

A public value, h , is computed using (7) and (9) as follows:

$$h = F_q \cdot g \bmod q. \quad (10)$$

Thus, CPKC has the private (secret) key, $SK = (f, g, q, F_g, F_q)$, and the public key, $PK = (h, q)$. The plaintext message, m , meets the following condition:

$$0 < m < \sqrt{q/4}. \quad (11)$$

A random integer, r , is chosen as follows:

$$0 < r < \sqrt{q/2}. \quad (12)$$

3.3.1. CPKC Encryption

The ciphertext, e , is computed using (10)–(12) as follows:

$$e = F_h(m, r) = r \cdot h + m \bmod q. \quad (13)$$

3.3.2. CPKC Decryption

Decryption is described by Steps 1 and 2 below:

Step 1: Multiply the ciphertext (13) by f , getting:

$$\begin{aligned} a &= f \cdot e \bmod q \\ &= r \cdot f \cdot F_q \cdot g + f \cdot m \bmod q. \end{aligned} \quad (14)$$

Note that $a = r \cdot g + f \cdot m$ if (the remainder is allowed to be negative):

$$|r \cdot g + f \cdot m| < q, \quad (15)$$

where (9), (10) and (13) are used. The CPKC decryption correctness condition (15) holds under Conditions (7), (11) and (12):

$$|r \cdot g + f \cdot m| < \sqrt{q/2}\sqrt{q/2} + \sqrt{q/2}\sqrt{q/4} < q.$$

Thus, the parameters, f , g , and r , are selected small compared to q (see (7), (11) and (12)) to meet the CPKC correctness decryption condition (15) used in Step 2 of the decryption.

Step 2: Multiply (14) by F_g , getting:

$$m = a \cdot F_g \bmod g, \quad (16)$$

where (9) is used and the contributor with the factor g in (14) vanishes due to (15). Numerical Example S1 of CPKC encryption/decryption is in the Supplementary Materials.

3.4. Formulas for CPU Power Consumption Calculation

Power, P , and energy, E , are measured in watts (W) and joules (J) [44], respectively, and calculated as follows:

$$P = V \cdot I, \quad (17)$$

$$E = P \cdot T, \quad (18)$$

where V is the potential difference measured in volts (V), I is the electric current measured in amperes (A), and T is the running time in seconds. There are three contributors to the CPU power consumption: dynamic, short-circuit, and power loss due to transistor leakage currents [45]:

$$P_{cpu} = P_{dyn} + P_{sc} + P_{leak}. \quad (19)$$

Power consumption is mainly defined by the dynamic and leakage components [46]. Leakage power, caused by leakage currents, is present in any active circuit independent of clock rates and is calculated as follows [46]:

$$P_{leak} = V \cdot I_{leak}, \quad (20)$$

where V is the supply voltage and I_{leak} is the leakage current. Dynamic power consumption depends on circuit activity (i.e., transistor switches, changes of values in registers, etc.) and is defined as follows [45]:

$$P_{dyn} = a \cdot C \cdot V^2 \cdot f, \quad (21)$$

where a is the switching activity factor, C is the capacitance measured in farad (F), and f is the clock frequency measured in hertz (Hz). Mostly, the activity factor is $a = 0.5$ [47]. MSP430FR5969, a Texas Instruments microcontroller with capacitance $C = 20$ pF [48] (Tables 5–12), active supply voltage from 1.8, ..., 3.6 V [48] (p. 1), clock frequency from 1, ..., 16 MHz [48] (p. 19), $I_{leak} = 20$ nA [48] (Tables 5–11), is used for RCPKC power consumption evaluation in Section 8.2.

4. Analysis of LBRA Attack Against CPKC

In this section, LBRA using GLR against the CPKC private key/message is described. Our implementation of GLR attack is shown (Maple 2016.2 is used throughout the paper). A demonstration by an example of how the CPKC private key can be attacked using GLR is presented. Then, a region defined in terms of Minkowski's second theorem where GLR attack fails is shown.

4.1. Lattice Basis Reduction Attack by GLR on CPKC Private Key/Message

In the following, $\|x\|$, $(x \cdot y)$, $\lfloor a \rfloor$, and \mathbb{R} , denote the Euclidean norm [49] of the vector x , the dot product of the vectors, x and y , the rounding of the real number, a , and the set of real numbers, respectively.

Let $E(V_1, V_2) \subset \mathbb{R}^2$ be a two-dimensional lattice with basis vectors, V_1 and V_2 :

$$E(V_1, V_2) = \{a_1 V_1 + a_2 V_2 : a_1, a_2 \in \mathbb{Z}\}. \quad (22)$$

The GLR algorithm [26] (p. 437), shown in Code 1, upon termination returns the shortest vector v_1 in $E(V_1, V_2)$.

Code 1. GLR algorithm pseudocode finding the shortest vector v_1 of the lattice $E(V_1, V_2)$.

Input: basis vectors V_1, V_2 ;

Output: the shortest vector v_1 in $E(V_1, V_2)$;

$v_1 = V_1; v_2 = V_2$;

Loop

If $\|v_2\| < \|v_1\|$

 swap v_1 and v_2 .

 Compute $m = \lfloor (v_1 \cdot v_2) / \|v_1\|^2 \rfloor$.

 If $m = 0$

 return the shortest vector v_1 of the basis, $\{v_1, v_2\}$.

 Replace v_2 with $v_2 - mv_1$.

End Loop.

The CPKC private key recovery problem can be formulated as the shortest vector problem (SVP) in the two-dimensional lattice, $E(V_1, V_2)$. From (10), it can be noticed that for any pair of integers, F and G , satisfying:

$$G = Fh \bmod q, \quad F = \mathcal{O}(\sqrt{q}), \quad G = \mathcal{O}(\sqrt{q}), \quad (23)$$

(F, G) is likely to serve as the first two components, f and g , of the private key, SK [26] (p. 376). Equation (23) can be written as $F \cdot h + q \cdot n = G$, where n is an integer. Therefore, our task is to find a pair of comparatively small by absolute value integers, (F, G) , such that:

$$F \cdot V_1 + n \cdot V_2 = (F, G), \quad (24)$$

where $V_1 = (1, h)$ and $V_2 = (0, q)$ are basis vectors, at least one of them having the Euclidean norm of order $\mathcal{O}(q)$. Similarly, the CPKC message recovery problem can be formulated as the SVP in the two-dimensional lattice, $E(V_1, V_2)$, where V_1 and V_2 are from (24). It can be also noticed from (13) that for any pair of integers, (RR, EM) , satisfying:

$$EM = RR \cdot h \bmod q, \quad RR = \mathcal{O}(\sqrt{q}), \quad EM = \mathcal{O}(\sqrt{q}), \quad (25)$$

(RR, EM) is likely to serve as the vector $(r, e - m)$ since the encryption Equation (13) can be written as $r \cdot h + q \cdot n = e - m$, where n is an integer. Therefore, our task is to find a pair of comparatively small by absolute value integers, (RR, RM) , such that:

$$RR \cdot V_1 + n \cdot V_2 = (RR, EM). \quad (26)$$

Our aim is to find the shortest vector w from $E(V_1, V_2)$ using GLR that might disclose $(r, e - m)$ if e and r are of the order of $\mathcal{O}(\sqrt{q})$. Comparing (24) and (26), it is noticed that they are the same up to the unknowns' names used, and hence, finding the shortest vector in $E(V_1, V_2)$ may reveal either the private key components, $(F, G) = (f, g)$, or the message related vector, $(RR, EM) = (r, e - m)$.

Code S1 in the Supplementary Materials presents our Maple [50] implementation of LBRA by GLR based on Code 1. Example S2 in the Supplementary Materials provides an example of LBRA attack using GLR against CPKC.

This section concludes that CPKC can be easily attacked using GLR. In order to modify CPKC to become resistant to GLR attack, first, in Section 4.2, a region where GLR attack fails is shown.

4.2. Region Resistant to GLR Attack on the CPKC Private Key/Message

LBRA by GLR succeeds in finding the CPKC private key, since it, by using the settings (7), is likely the shortest vector in the lattice. Minkowski's second theorem [51] (p. 35) sets an upper bound for the norm of the shortest nonzero vector, λ , in a two-dimensional lattice:

$$\lambda \leq \sqrt{\lambda_2} \text{Vol}(L)^{1/2}, \quad (27)$$

where $\lambda_2 = 2/\sqrt{3} \approx 1.154$ is Hermite's constant [51] (p. 41) and $\text{Vol}(L)$ is the volume of the lattice, which is equal to q for the lattice $L = E(V_1, V_2)$ where V_1 and V_2 are defined in (24). Therefore, (27) can be written as follows:

$$\lambda \leq \alpha \sqrt{q}, \quad (28)$$

where $\alpha = \sqrt{\lambda_2} \approx 1.07$. From (28), one gets for the relative norm,

$$\lambda' = \frac{\lambda}{\sqrt{q}}, \quad (29)$$

the following inequality (30):

$$\lambda' \leq \alpha. \quad (30)$$

GLR fails in attacking the CPKC private key/message when (30) is not satisfied for the secret vector relative norm (f, g) , i.e., if:

$$\|(f, g)\|/\sqrt{q} > \alpha \quad (31)$$

holds, GLR fails to find the CPKC private key/message.

CPKC selects small values for private key (f, g) in (7) to satisfy the decryption correctness condition (15). Hence, our goal is to propose in Section 5 a modification for CPKC, that is RCPKC, where (f, g) satisfies (31) and provides correct decryption for valid users and incorrect decryption for an attacker using GLR.

5. The Proposed RCPKC

In this section, random CPKC (RCPKC), an adjustment of CPKC described in Section 3.3, so that it becomes resistant to GLR attack, is proposed.

5.1. RCPKC's Main Ideas

The main two ideas of RCPKC are:

- Contrary to the settings (7) of CPKC, which uses secret key (f, g) with a small norm not exceeding \sqrt{q} so that (f, g) may be found as a shortest vector (SV) in the lattice $E(V_1, V_2)$ defined by (24), RCPKC [42] (we call it in this section RCPKC.1) was originally proposed having private key (f, g) with a large norm meeting (31) so that it cannot be returned by LBRA using GLR as the SV, but (f, g) also meets (15) due to the skew in its components.
- However, as mentioned in Section 4.1, for any pair of integers, F and G , satisfying (23), (F, G) is likely to serve as the first two components, f, g , of the private key. That means, in spite of the large norm of (f, g) , the $SV = (F, G)$, obtained in the result of LBRA using GLR, may meet the decryption correctness condition (15) and, thus, may be used for the correct plaintext message disclosure (Example S4 shows the LBRA attack using GLR against RCPKC.1; see the Supplementary Materials). That is why RCPKC.1, Section 5.2, before encrypting by (13) (contrary to CPKC using a random number from the predefined range (12)), defines a range for the random number selection using the SV, (F, G) (returned by GLR attack on the lattice $E(V_1, V_2)$ defined by (24)), so that the decryption correctness condition (15) holds for (f, g) , but does not hold for (F, G) , which leads to the failure of LBRA using GLR on RCPKC.1. Such an interval defined in (40)–(42) for RCPKC.1 is found to be vulnerable to the GLR attack. Therefore, an enhanced RCPKC proposed herein (we call it in this section RCPKC.2) with a tighter interval for r is defined in (46), (50) and (51), so that such an attack is inactive.

Thus, RCPKC.2 assumes that the private key owner selects a range for a random value, r (used in encryption (13)), based on the secret key, (f, g) , and respective SV, (F, G) , in the lattice, $E(V_1, V_2)$, defined by (24), guaranteeing correct decryption for a valid user and incorrect decryption for an attacker using GLR. Because of the special choice of the random value range, the proposed algorithm is called random CPKC (RCPKC). The problem for RCPKC is that the range of random numbers as so defined may be rather narrow, and thus, the security of RCPKC may suffer. However, as will be shown, the range is rather large and may significantly exceed the range for a secret message.

In Section 5.2, CPKC is modified to RCPKC.1 [42] so that the secret key, (f, g) , meets (15) and (31). In Section 5.3, RCPKC.1 is further modified to RCPKC.2, so that it becomes immune to the LBRA attack.

5.2. RCPKC.1 Description

To meet (31), it is required that:

$$f, r \geq \alpha \cdot \sqrt{q}. \quad (32)$$

The LBRA by GLR failure condition (31) holds if (32) is true since:

$$\begin{aligned} \frac{\|(f, g)\|}{\sqrt{q}} &= \frac{\sqrt{f^2 + g^2}}{\sqrt{q}} = \frac{\sqrt{\alpha^2 \cdot q + g^2}}{\sqrt{q}} > \alpha, \\ \frac{\|(r, e - m)\|}{\sqrt{q}} &= \frac{\sqrt{r^2 + (e - m)^2}}{\sqrt{q}} = \frac{\sqrt{\alpha^2 \cdot q + (e - m)^2}}{\sqrt{q}} > \alpha, \end{aligned}$$

for $g, e - m \neq 0$. Condition (32), in RCPKC.1, substitutes the conditions (7) and (12) on f and r in CPKC. The message, m , and the private key, g , instead of (11) and (7), used in CPKC, are redefined in RCPKC.1 as follows:

$$2^{mgLen} > g \geq 2^{mgLen-1} > m \geq 0, \quad (33)$$

where $mgLen$ represents the length of m and g in bits.

For RCPKC.1, the correctness decryption condition (15) shall hold, which is true (see (39)) when the f and r values in addition to (32) meet (34):

$$\frac{q}{2 \cdot 2^{mgLen}} > f, r. \quad (34)$$

Since:

$$q = 2^{qLen}, \quad (35)$$

then (32) and (34) can be rewritten:

$$2^{qLen-mgLen-1} > f, r \geq \alpha \cdot 2^{qLen/2}. \quad (36)$$

To have a non-empty range for f and r , of a width of at least $\alpha \cdot 2^{qLen/2}$, from (36), the following condition is obtained:

$$\frac{2^{qLen/2}}{2 \cdot \alpha} > 2^{mgLen+1}. \quad (37)$$

By defining $\beta = \log_2 1/(2 \cdot \alpha) \approx -1.103$, (37) shows that:

$$\begin{aligned} 2^\beta \cdot 2^{qLen/2} &> 2^{mgLen+1}, \\ qLen + 2 \cdot \beta &> 2 \cdot (mgLen + 1), \\ qLen &> 2 \cdot (mgLen + 1 - \beta). \end{aligned} \quad (38)$$

Let us show that the decryption correctness condition (15) holds when (33), (36) and (38) hold:

$$\begin{aligned} r \cdot g + f \cdot m &< 2^{qLen-mgLen-1} \cdot 2^{mgLen} + 2^{qLen-mgLen-1} \cdot 2^{mgLen-1} \\ &< 2^{qLen-1} + 2^{qLen-1} = 2^{qLen} = q. \end{aligned} \quad (39)$$

Thus, for RCPKC.1, the norm of (f, g) meets (31), and the decryption correctness condition (39) holds. We need additionally that decryption correctness condition (39) to be violated for (F, G) , that is the SV obtained in the result of the GLR attack on the lattice $E(V_1, V_2)$ defined by (24). Hence, it cannot be used as a private key for the plaintext message's correct decryption.

Inequality (36) defines a range for r so that f, g, r, m meet (15). Now, we define constant on r ,

$$r \geq rmin \geq (q + g|F|)/|G| \quad (40)$$

such that F, G, r, m violate (15). Using (40) and (33):

$$\begin{aligned} |G \cdot r + F \cdot m| &\geq |G| \cdot |r| - |F| \cdot m \geq \frac{|G|(q + g|F|)}{|G|} - |F| \cdot m \\ &\geq q + g|F| - m|F| > q. \end{aligned} \quad (41)$$

Thus, Inequality (36) is used for f , but for r , from (40) and (36), we have:

$$2^{qLen-mgLen-1} > r \geq \max(\alpha \cdot 2^{qLen/2}, rmin). \quad (42)$$

For RCPKC security, the range defined by (42) shall be rather large, $\max(\alpha \cdot 2^{qLen/2}, rmin)$; hence:

$$2^{qLen-mgLen-1} \geq 2 \cdot \max(\alpha \cdot 2^{qLen/2}, rmin). \quad (43)$$

Thus, RCPKC.1 is defined as follows.

RCPKC.1 definition:

The private key components, (f, g) , meet (8), (9), (33) and (34), where $qLen, mgLen$ meet (38) and (43), where (F, G) is an SV obtained in the result of the GLR attack on the lattice $E(V_1, V_2)$ defined by (24). The public key component, h , is defined by (10). Message, m , meets (33), and random integer, r , is selected from the range defined in (40) and (42). Encryption and decryption follow (13) and (14), (16), respectively (see Sections 3.3.1 and 3.3.2). The decryption correctness condition (15) is proven for

RCPKC.1 in (39). Examples S3 and S4 in the Supplementary Materials present an example of RCPKC.1 encryption/decryption and the LBRA attack using the GLR attack against RCPKC.1, respectively.

In the following section, RCPKC.1 is modified to RCPKC.2, so that it becomes immune against the LBRA attack.

5.3. RCPKC.2 Proposal

In order to resist the GLR attack against RCPKC.1, as shown in Example S3, the definition of the region from which r is selected should consider all SVs with a norm less than a threshold $\mu\|(f, g)\|$ as follows.

The random interval defined in (40), (42) and (43) using only the SV obtained by the GLR attack on the lattice $E(V_1, V_2)$ defined by (24) must be modified to include all the SVs with a norm less than the norm of the secret key, by threshold $\mu\|(f, g)\|$. Hence, all vectors (F_i, G_i) obtained in the course of GLR reduction that have norms:

$$\|(F_i, G_i)\| < \mu\|(f, g)\|, i = 1, \dots, N, \quad (44)$$

where N is the number of (F, G) pairs satisfying (44), μ is a threshold, e.g., $\mu = 10$, and then, it must be checked that:

$$(\forall i = 1, \dots, N)((F_i, G_i) \neq (f, g)). \quad (45)$$

If (45) is violated, i.e., one of the vectors in the list is our vector (f, g) , then another (f, g) is used.

Inequality (36) defines a range for r so that f, g, r, m meet (15). Now, the constraint on r is defined as follows:

$$q/g - f \geq rmax \geq r \geq rmin \geq (q + g \cdot \max_{i=1, \dots, N} |F_i|) / \min_{i=1, \dots, N} |G_i|, \quad (46)$$

such that F_j, G_j, r, m violate (15) for any $j = 1, \dots, N$. We require also that:

$$h \cdot rmin > q. \quad (47)$$

Using (33) and (46), it is noticed that actually, the decryption correctness condition (15) for any $j = 1, \dots, N$, is violated:

$$\begin{aligned} |G_j \cdot r + F_j \cdot m| &\geq |G_j \cdot r| - |F_j \cdot m| \geq |G_j| \cdot \frac{q + g \cdot \max_{i=1, \dots, N} |F_i|}{\min_{i=1, \dots, N} |G_i|} - |F_j \cdot m| \\ &\geq q + g \cdot \max_{i=1, \dots, N} |F_i| - |F_j \cdot m| > q. \end{aligned} \quad (48)$$

From (33) and (46), it is also perceived that the decryption correctness condition (15) holds for the original (f, g) :

$$g \cdot rmax + f \cdot m \leq g(q/g - f) + f \cdot m = q - f \cdot g + f \cdot m < q \quad (49)$$

Thus, Inequality (36) is used for f , but for r from (36) and (46):

$$rmax > r \geq \max(\alpha \cdot 2^{qlen/2}, rmin). \quad (50)$$

For RCPKC.2's security, the range defined by (50) shall be rather large, such as, e.g., $\max(\alpha \cdot 2^{qlen/2}, rmin)$; hence, it is desirable to have:

$$rmax \geq 2 \cdot \max(\alpha \cdot 2^{qlen/2}, rmin). \quad (51)$$

In order to provide CCA indistinguishability (see Definition 2 and Section 7), it is required to have:

$$\gcd(g, q) > 1. \quad (52)$$

Thus, the RCPKC.2 proposal follows.

RCPKC.2 proposal:

The private key components, (f, g) , meet (8), (9), (33), (34) and (52), where $qLen, mgLen$ meet (38) and (43). The public key component, h , is defined by (10). Message, m , meets (33), and random integer, r , is selected from the range defined in (46), (47) and (50). Encryption and decryption follow (13), (14) and (16), respectively (see Sections 3.3.1 and 3.3.2). The decryption correctness condition (15) is proven for RCPKC in (49). Example S5 in the Supplementary Materials presents an example of finding RCPKC.2's random interval and LBRA by GLR failure.

RCPKC.2 is more secure than RCPKC.1 because intermediate GLR outputs are also used for the random parameter range selection. However, their computational complexity is the same, since both employ GLR and follow the same encryption/decryption procedures.

RCPKC.2 is also resistant to various attacks, as shown in the security analysis presented in the next section. Note that hereafter, RCPKC.2 is again denoted as RCPKC.

6. RCPKC Security Analysis

In this section, attacks on NTRU are considered (brute force (on the key and message), meet-in-the-middle (MITM) in Section 6.1, lattice basis reduction in Section 6.4, hybrid lattice basis reduction and MITM [52] in Section 6.2, multiple transmission (MTA) [11] in Section 6.3, and also, the most recent, chosen ciphertext [53–56], in Section 6.5), and we try applying them to RCPKC. Herein, the NTRU parameters used, EES401EP1 [41], of the security level, $k = 112$ bits:

$$\begin{aligned} N &= 401, p = 3, q = 2048, df_1 = df_2 = 8, \\ df_3 &= 6, dg = 133, dr_1 = dr_2 = 8, dr_3 = 6. \end{aligned} \quad (53)$$

In order to meet the same security level, the RCPKC settings satisfying (38) are:

$$qLen = 473, mgLen = 225. \quad (54)$$

The key space cardinality (defined in Section 6.1 for the parameters (53) and (54)) is greater than or equal to $2^{2 \cdot k}$ for $k = 112$ to avoid the MITM attack explained in Section 6.1.

6.1. Brute Force and MITM Attacks

An attacker can recover the NTRU private key by trying all possible values of g and testing whether $f \cdot h \bmod q$ has small coefficients (the product corresponds to g according to (10)). On the other hand, an attacker can try all possible values of g and test whether $h^{-1} \cdot g \bmod q$ (corresponding to f by virtue of (10)) has small coefficients. Equations (55) and (56) show the search space cardinalities for g and f for the security level, $k = 112$ (taking into account the MITM attack explained later in this section). The search space cardinality for f is computed as follows (see [53] (Section 7)):

$$\begin{aligned} C_{NTRU}(f, k) &= \binom{N}{df_1} \binom{N - df_1}{df_1} \binom{N}{df_2} \binom{N - df_2}{df_2} \binom{N}{df_3} \binom{N - df_3}{df_3} \\ &= \binom{401}{8} \binom{393}{8} \binom{401}{8} \binom{393}{8} \binom{401}{6} \binom{395}{6} \\ &= 1.16 \times 10^{90} \geq 2^{2 \cdot k} = 2^{224}. \end{aligned} \quad (55)$$

Similarly, for g :

$$\begin{aligned} C_{NTRU}(g, k) &= \binom{N}{dg} \binom{N - dg}{dg} \\ &= \binom{401}{133} \binom{268}{133} = 4.34 \times 10^{188} \geq 2^{2 \cdot k} = 2^{224}. \end{aligned} \quad (56)$$

it is perceived the search space cardinality for f is less than that for g , so the best strategy for an attacker is to search for f values.

An attacker can reduce the search space cardinality from 2^k to $2^{k/2}$ [57] using MITM by splitting the private key f (which is a polynomial of degree $N - 1$) into two polynomials, $f = f_1 + f_2$, where f_1 is a polynomial of degree at most $N/2 - 1$ and polynomial f_2 contains terms of degree between $N/2$ and $N - 1$, and then trying matches: $f_1 \cdot h \bmod q = (g - f_2 \cdot h) \bmod q$. Hence, in order to meet the $k = 112$ security level, the NTRU parameters must be chosen to meet the $k = 224$ security level, as it is already made in (53). For RCPKC, the secret value, g , is selected from the interval $[2^{mgLen-1}, 2^{mgLen})$ (see (33)); hence, the search space cardinality for g to meet the $2 \cdot k$ -bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(g, k) = 2^{mgLen-1} \geq 2^{2 \cdot k}. \quad (57)$$

The secret value, f , is selected from the interval $[\alpha \cdot 2^{qLen/2}, 2^{qLen-mgLen-1})$ (see (36)); hence, the search space cardinality for f to meet the $2 \cdot k$ -bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(f, k) = 2^{qLen-mgLen-1} - \alpha \cdot 2^{qLen/2} \geq 2^{2 \cdot k}. \quad (58)$$

For the parameters (54), $C_{RCPKC}(g, k) = 2^{224}$, while $C_{RCPKC}(f, k) \approx 2^{247}$. In order to provide the security level for $k = 112$, the parameters (54) are chosen to meet the twice greater security level of $2 \cdot k = 224$ to counter the MITM attack, considered below, which reduces the brute force attack effort by the square root. Since $C_{RCPKC}(g, k) < C_{RCPKC}(f, k)$, the best strategy for an attacker is to search for g values. Similar to NTRU, the MITM attack can be applied to the RCPKC private key component, g . Since $mgLen$ is the bit length of g , then $g = g_1 + 2^{(mgLen-1)/2}g_2$, and then, g_1 and g_2 , each of a bit length equal to $(mgLen - 1)/2$, can be enumerated with the resulting search space cardinality $\mathcal{O}(2^{(mgLen-1)/2})$ trying to find matching:

$$(f \cdot h - g_1) \bmod q = 2^{(mgLen-1)/2}g_2 \bmod q.$$

Thus, the RCPKC parameters (54) provide the security level $k = 112$ against the brute force attack with MITM. Now, let us consider the brute force attack on the message.

An attacker can compromise an NTRU message by trying all possible values of r and testing whether $e - r \cdot h \bmod q$ has small coefficients. Similarly, the attacker can compromise the RCPKC message by trying all possible values of r and testing if $e - r \cdot h \bmod q \in [0, 2^{mgLen-1})$ by virtue of (33).

The RCPKC message search space is defined by the interval $[0, 2^{mgLen-1})$ (see (33)); hence, the search space cardinality for m to meet the $2 \cdot k$ -bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(m, k) = 2^{mgLen-1} \geq 2^{2 \cdot k}, \quad (59)$$

while the search space of r is defined by (46), (50) and (51). Hence, the search space cardinality for r to meet the $2 \cdot k$ -bit security level against the brute force attack shall satisfy:

$$C_{RCPKC}(r, k) = rmax - \max(\alpha \cdot 2^{qLen/2}, rmin) \geq 2^{2 \cdot k}. \quad (60)$$

Table 1 shows the $mgLen$ and $qLen$ values to meet different $2 \cdot k$ -bit security levels' condition (60) (see Rows 1 and 2) and the width of the range for r (Row 7) with f and g specified in Rows 3 and 4, respectively. It proves that the method can be practically used.

Table 1. Width of the range for the r value for different security levels (Row 7); the parameters of the random congruential public key cryptosystem (RCPKC) affecting the width ($mgLen, qLen, f, g, rmax, \max(\alpha \cdot 2^{qLen/2}, rmin)$) are specified in Rows 1–6.

#	RCPKC Parameter	$2 \times k$		
		224	336	448
1	$mgLen$	225	337	450
2	$qLen$	473	743	909
3	$f = 2^{qLen-mgLen-1} - 1$	2.26×10^{74}	8.26×10^{121}	7.44×10^{137}
4	g	$2^{mgLen} - 1 = 5.39 \times 10^{67}$	$2^{mgLen} - 5 = 2.79 \times 10^{101}$	$2^{mgLen} - 11 = 2.90 \times 10^{135}$
5	$rmax$	2.26×10^{74}	8.26×10^{121}	7.44×10^{137}
6	$\max(\alpha \cdot 2^{qLen/2}, rmin)$	7.41×10^{72}	1.62×10^{119}	1.10×10^{137}
7	$C_{RCPKC}(r, k)$	2.1×10^{74}	8.24×10^{121}	6.34×10^{137}

6.2. A Hybrid Lattice Basis Reduction and MITM Attack

The attack [52] on the NTRU secret key combines the LBRA and MITM strategies. The hybrid attack, first, splits the original lattice of order $2N$, $N > 1$, into three subparts, only one of which is further reduced, whereas the vectors from the other parts are just enumerated, thus combining the concepts of the LBRA and MITM attacks. The hybrid attack is not applicable to RCPKC since:

- The RCPKC lattice is two-dimensional and cannot be split into the three subparts;
- RCPKC uses a large norm secret (f, g) vector (see (33) and (36)) that cannot be found by LBRA looking for an SV, and the SV cannot be used for correct decryption (see (48)).

6.3. Multiple Transmission Attack

MTA reveals a large part of an NTRU message by sending n times one and the same message, m , using the same public key, h , but different random values, r^i . For NTRU encryption (13) (see Section 3.1):

$$e^i = r^i \cdot h + m \bmod q$$

for $i = 1, 2, \dots, n$. An adversary computes:

$$(e^i - e^1) \cdot h^{-1} \bmod q,$$

thereby recovering $r^i - r^1 \bmod q$, $i = 1, \dots, n$, and from these relations, many coefficients of r^1 may be revealed. Knowledge of r^1 allows disclosing the message, m . RCPKC is not susceptible to MTA because no special structure is assumed for r^1 contrary to the case of NTRU.

6.4. Lattice Basis Reduction Attacks

The NTRU lattice basis, L_h^{NTRU} , associated with public key h defined in (1) is:

$$L_h^{NTRU} = \left(\begin{array}{cccc|cccc} 1 & 0 & \dots & 0 & h_0 & h_1 & \dots & h_{N-1} \\ 0 & 1 & \dots & 0 & h_{N-1} & h_0 & \dots & h_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 & h_1 & h_2 & \dots & h_0 \\ \hline 0 & 0 & \dots & 0 & q & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & q \end{array} \right),$$

where h_0, \dots, h_{N-1} are the coefficients of the polynomial h . For convenience, matrix L_h^{NTRU} is abbreviated as:

$$L_h^{NTRU} = \begin{pmatrix} I & h \\ 0 & qI \end{pmatrix}.$$

The NTRU private key recovery problem can be formulated as the SVP in $2N$ -dimensional lattice, L_h^{NTRU} . Actually, if a polynomial, b , of degree $N - 1$ with integer coefficients satisfying:

$$f \cdot h + q \cdot b = g$$

exists, then:

$$(f, b) \cdot L_h^{NTRU} = (f, g).$$

Therefore, the vector (f, g) is in the lattice L_h^{NTRU} . Vector (f, g) or its rotation (rotation of a polynomial, f , by i steps is $x^i \cdot f \in R_q$ for an integer i) can be found if it is the shortest vector in L_h^{NTRU} . The lattice reduction algorithm LLL [51] finds the shortest vector in L_h^{NTRU} in time exponential in N . According to [40], LLL takes 1.05×10^{31} MIPS (million instructions per second)-years to find the shortest vector or its rotation for $N = 400$ (as in (53)) that most likely is the NTRU private key part, (f, g) .

Contrary to NTRU, RCPKC is resistant to LBRA since the GLR attack fails for it (see Section 5). LBRA is one of the most used and effective techniques in attacking an NTRU private key (e.g., it is used in the hybrid lattice attack, the most efficient on practical NTRU parameters [58]; see Section 6.2), but it is not applicable to RCPKC.

6.5. Chosen Ciphertext Attack

Three chosen ciphertext attacks (CCA) on NTRU are known. The first key recovering CCA described in [54] uses a ciphertext of a special shape, which can be countered by message padding [53]. Standardized parameters [53] allow decryption failure, i.e., a ciphertext could fail to be decrypted correctly by NTRU. In [55], a CCA was presented where an attacker collects a large number of decryption failures; see the NTRU correction decryption condition (5) in Section 3.1. Another CCA was presented in [56], which is more efficient than [55], but still depends on decryption failures. RCPKC works on non-structured integers, and the parameters, set in Section 5, guarantee correct decryption. Thus, neither of the CCAs described above are applicable to RCPKC.

7. RCPKC Asymmetric Encryption Padding and its IND-CCA2 Security

In this section, we prove the security of the RCPKC one-way function based on the discussions of the security of the NTRU one-way function in [8], define RCPKC asymmetric encryption padding (RAEP), and prove its IND-CCA2 security as a particular case of NAEP. According to Sections 5.2 and 5.3, RCPKC defines the following four sets:

- $\mathcal{D}_f = [\alpha \cdot 2^{qLen/2}, 2^{qLen - mgLen - 1}]$: private key space, an interval from which a private key, f , is selected;
- $\mathcal{D}_g = [2^{mgLen - 1}, 2^{mgLen}]$: private key space, an interval from which a private key, g , is selected;
- $\mathcal{D}_m = [0, 2^{mgLen - 1}]$: RCPKC plaintext space, an interval from which a plaintext, m , is selected;
- $\mathcal{D}_r = [\max(\alpha \cdot 2^{qLen/2}, rmin), rmax]$: RCPKC random value space.

The RCPKC encryption primitive is specified by the parameter set, $\mathcal{P} = (q, \mathcal{D}_f, \mathcal{D}_g, \mathcal{D}_m, \mathcal{D}_r)$. The one-way function underlying RCPKC is:

$$F_h : \mathcal{D}_m \times \mathcal{D}_r \rightarrow \mathbb{Z}_q,$$

$$F_h(m, r) = r \cdot h + m \bmod q.$$

Definition 3. RCPKC-OW problem: For a parameter set, \mathcal{P} , we denote by $Succ_{RCPKC}^{OW}(\mathcal{A}, \mathcal{P})$ the success probability of a PPT adversary, \mathcal{A} , for finding a pre-image of F_h ,

$$Succ_{RCPKC}^{OW}(\mathcal{A}, \mathcal{P}) = \Pr \left(\begin{array}{l} (m', r') \leftarrow \mathcal{A}(e, h) \\ \text{s.t. } (\exists r' \in \mathcal{D}_r) (F_h(m', r') = e) \end{array} \right).$$

Assumption 2. RCPKC-OW assumption: For every PPT adversary, \mathcal{A} , solving the RCPKC-OW problem, there exists a negligible function, $v_A(k)$, such that for sufficiently large k , we have:

$$Succ_{RCPKC}^{OW}(\mathcal{A}, \mathcal{P}) \leq v_A(k).$$

An adversary \mathcal{A}_1 can compromise (m, r) by picking $r' \in \mathcal{D}_r$, substituting it in $(e - r' \cdot h) \bmod q$, and checking, if the result is in \mathcal{D}_m . Thus, $Succ_{RCPKC}^{OW}(\mathcal{A}_1, \mathcal{P})$ is:

$$Succ_{RCPKC}^{OW}(\mathcal{A}_1, \mathcal{P}) = \frac{2^{mgLen}}{2^{qLen}}.$$

Since $qLen > mgLen$ by definition (38), $Succ_{RCPKC}^{OW}(\mathcal{A}_1, \mathcal{P})$ decreases exponentially in $qLen$, and Assumption 1 holds. Similarly, the attacker can try the following methods with an exponentially decreasing success probability:

1. The adversary, \mathcal{A}_2 , chooses randomly a pair $(r' \in \mathcal{D}_r, m' \in \mathcal{D}_m)$ and checks if $r' \cdot h + m' \bmod q = e$.
2. The adversary, \mathcal{A}_3 , picks $f' \in \mathcal{D}_f$, substitutes it in $f' \cdot h \bmod q$, and checks whether the result is in \mathcal{D}_g .
3. The adversary, \mathcal{A}_4 , chooses randomly a pair $(f' \in \mathcal{D}_f, g' \in \mathcal{D}_g)$, if possible, calculates h' , decrypts e to (r', m') , and checks if $r' \cdot h' + m' \bmod q = e$.
4. Furthermore, the adversary can apply the GLR attack to get (f, g) . However, by construction, RCPKC is immune to that attack, and hence, the success probability is zero. Therefore, Assumption 1 is true for all the above attacks.

RCPKC encryption (13) differs from NTRU encryption (2) just by setting $N = p = 1$. The conclusion of [7] on NAEP IND-CCA2 security is also true for asymmetric encryption padding, RAEP. However, NAEP cannot be used as is for $N = p = 1$ because it utilizes specific true polynomial functions center() and compress(). Since the decryption correctness condition (15) holds for RCPKC

due to the parameter choice, the center() function is not used in RCPKC and RAEP. The function compress() as in NAEP shall map its input, $p \cdot r \cdot h$, to a binary string, bs , of the padded message size. In NAEP, it is done in two steps: $s = \text{compress}(p \cdot r \cdot h \bmod q)$; $bs = H(s)$. In RAEP, both transforms are done by one hash function, $H : \mathbb{Z}_q \rightarrow \{0, 1\}^{mgLen}$. Algorithms 3 and 4 show RAEP encryption and decryption, respectively.

Algorithm 3: RAEP encryption.

input : $N = mgLen = \theta(k)$ is the length of the RCPKC encrypted message; $N > l = \theta(k)$ is the padding size; $G : \{0, 1\}^{N-l} \times \{0, 1\}^l \rightarrow D_r$ and $H : \mathbb{Z}_q \rightarrow \{0, 1\}^N$ are hash functions; D_r is defined by (45), (49) and (50); $m \in \{0, 1\}^{N-l}$ is the input plaintext message.

output: $e \in \mathbb{Z}_q$ is the ciphertext.

begin

1. Pick $\mu \leftarrow_R \{0, 1\}^l$
2. Let $\rho = G(m, \mu)$, $r = \text{genr}(\rho)$, $s = r \cdot h \bmod q$, and $\omega = (m || \mu) \oplus H(s)$ // genr is a //function generating correct r according to (46), (47), (50) and (51)
3. Let $e = F_h(m, r)$ //according to (13)

end

Algorithm 4: RAEP decryption.

input : $N = \theta(k)$ is the length of the RCPKC encrypted message; $N > l = \theta(k)$ is the padding size; $G : \{0, 1\}^{N-l} \times \{0, 1\}^l \rightarrow D_r$ and $H : \{0, 1\}^N \rightarrow \{0, 1\}^N$ are hash functions; D_r is the space for r ; $e \in \mathbb{Z}_q$ is the ciphertext.

output: $m \in \{0, 1\}^{N-l}$ is the decrypted plaintext message if decrypted correctly, and Reject, otherwise.

begin

1. $a = f \cdot e \bmod q$ //according to (14),
2. $\omega = F_g \cdot a \bmod g$ //according to (16)
3. $s = e - \omega \bmod q$
4. $m || \mu = \omega \oplus H(s)$; $r = \text{genr}(G(m || \mu))$.
5. if $r \cdot h = s \bmod q$, then output m , else, output Reject.

end

8. RCPKC Performance and Power Consumption Evaluation

8.1. RCPKC Performance Evaluation

Experiments were conducted using the NTRU code [59] and RCPKC implementation in the C99 language similar to [59] with the NTL library [60] on a PC equipped with 1.6 GHz Intel Core i5-8250U, 8 GB RAM, and Windows 10 (see Tables S1 and S2 of the Supplementary Material for the RCPKC performance experiments' results and the NTRU performance experiments' results, respectively; the RCPKC source code is available in [61]). Both the NTRU code [59] and the proposed RCPKC were implemented in Visual Studio 2017. The NTRU parameters (53) and the RCPKC parameters (54) were used. The CPU encryption and decryption time of RCPKC and NTRU was measured for 10^3 , 10^4 , and 10^5 runs. In each run, a distinct 128 bit message was encrypted/decrypted with both cryptosystems. The NTL function RandomLen() was used to pseudo-randomly generate the messages. RandomLen() was seeded with the output of the function clock(). The generated messages were stored in a separate file and used to test RCPKC and NTRU. The CPU time was measured via QueryPerformanceCounter() with ns accuracy. Table 2 shows the sample mean, \bar{x} , standard deviation, σ , and confidence interval with the confidence level $C \in \{0.95, 0.99, 0.999\}$ for the number of runs $n \in \{10^3, 10^4, 10^5\}$, respectively for RCPKC and NTRU. The confidence interval, $[l, u]$, is calculated using [62] (p. 358):

$$[l, u] = \left[\bar{x} - z^* \frac{\sigma}{\sqrt{n}}, \bar{x} + z^* \frac{\sigma}{\sqrt{n}} \right], \quad (61)$$

where $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$, $\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$, x_i , and n are the sample mean, sample standard deviation, value of the run, and number of runs, respectively; z^* is the critical value required for the specific confidence level; see Table C in [62] (p. 746). For example, in Table 2 for RCPKC encryption with $C = 95\%$, $n = 10^3$, $\bar{x} = 6.19 \times 10^{-6}$, $\sigma = 3.966 \times 10^{-6}$, $z^* = 1.960$, the confidence interval is calculated as follows: $[l, u] = (6.190 \times 10^{-6} - 1.960(3.966 \times 10^{-6})/\sqrt{1000}, 6.190 \times 10^{-6} + 1.960(3.966 \times 10^{-6})/\sqrt{1000}) = (6.112 \times 10^{-6}, 6.267 \times 10^{-6})$.

Figure 1 shows the NTRU/RCPKC encryption and decryption average CPU time ratio for 10^3 , 10^4 , and 10^5 runs. From Figure 1, it is observed that RCPKC is 27.08 ± 3.75 times faster than NTRU in encryption and 26.9 ± 5.09 times faster in decryption, respectively. Table 3 compares NTRU versus RCPKC and several NTRU variants presented in Section 1. It is observed that RCPKC is faster than the fastest most recently published NTRU variant, BQTRU, more than four times in encryption.

Table 2. RCPKC and NTRU CPU encryption/decryption time sample mean, standard deviation, and confidence interval for 10^3 , 10^4 , and 10^5 runs.

Algorithm	Measured Value	Run number, n		
		10^3	10^4	10^5
	C	0.95	0.99	0.999
	z^*	1.960	2.576	3.291
RCPKC (Encryption)	Sample Mean, \bar{x}	6.190×10^{-6}	5.492×10^{-6}	4.708×10^{-6}
	Sample Standard deviation, σ	3.966×10^{-6}	2.076×10^{-6}	2.923×10^{-6}
	Confidence Interval, $[l, u]$	$(6.112 \times 10^{-6}, 6.267 \times 10^{-6})$	$(5.475 \times 10^{-6}, 5.508 \times 10^{-6})$	$(4.677 \times 10^{-6}, 4.738 \times 10^{-6})$
NTRU (Encryption)	Sample Mean, \bar{x}	1.444×10^{-4}	1.964×10^{-4}	1.440×10^{-4}
	Sample Standard deviation, σ	6.878×10^{-5}	1.123×10^{-5}	6.437×10^{-5}
	Confidence Interval, $[l, u]$	$(1.430 \times 10^{-4}, 1.457 \times 10^{-4})$	$(1.430 \times 10^{-4}, 1.973 \times 10^{-4})$	$(1.430 \times 10^{-4}, 1.447 \times 10^{-4})$
RCPKC (Decryption)	Sample Mean, \bar{x}	9.506×10^{-6}	8.812×10^{-6}	7.493×10^{-6}
	Sample Standard deviation, σ	2.781×10^{-6}	2.370×10^{-6}	2.795×10^{-6}
	Confidence Interval, $[l, u]$	$(9.451 \times 10^{-6}, 9.560 \times 10^{-6})$	$(8.792 \times 10^{-6}, 8.831 \times 10^{-6})$	$(7.464 \times 10^{-6}, 7.522 \times 10^{-6})$
NTRU (Decryption)	Sample Mean, \bar{x}	2.079×10^{-4}	2.826×10^{-4}	2.088×10^{-4}
	Sample Standard deviation, σ	9.700×10^{-5}	1.594×10^{-4}	8.633×10^{-5}
	Confidence Interval, $[l, u]$	$(2.060 \times 10^{-4}, 2.098 \times 10^{-4})$	$(2.813 \times 10^{-4}, 2.839 \times 10^{-4})$	$(2.079 \times 10^{-4}, 2.097 \times 10^{-4})$

Table 3. Ratios of encryption and decryption times, $\gamma_A = \frac{T_{NTRU}^{ENC}}{T_A^{ENC}}$, $\delta_A = \frac{T_{NTRU}^{DEC}}{T_A^{DEC}}$, of NTRU and the algorithms $A \in \{RCPKC, BQTRU, MaTRU, ETRU\}$.

Algorithm, A	Encryption Times Ratio, γ_A	Decryption Times Ratio, δ_A
Proposed RCPKC	27	27
BQTRU [21]	7	No data
MaTRU [23]	2.5	2.5
ETRU [27]	1.45	1.72

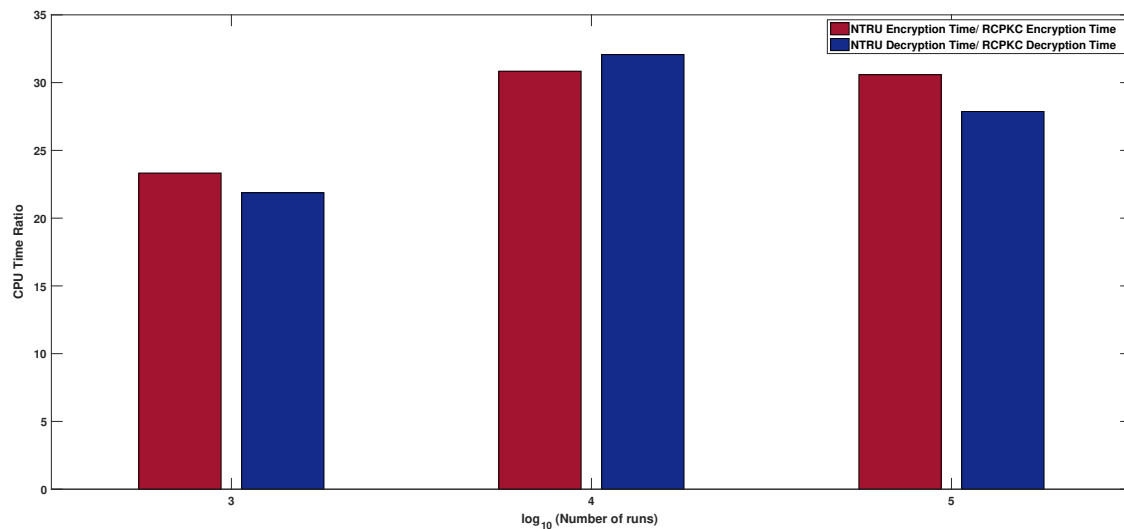


Figure 1. NTRU/RCPKC encryption and decryption average CPU time ratio for 10^3 , 10^4 , and 10^5 runs.

8.2. RCPKC Power Consumption Evaluation

In this section, RCPKC's power consumption is compared to NTRU in two cases: applying both algorithms using the same or different frequencies.

Same frequencies: Let the RCPKC and NTRU execution time be T_{RCPKC} and T_{NTRU} , respectively. Then, from (18), the consumed energy by NTRU and RCPKC E_{NTRU} and E_{RCPKC} is:

$$\begin{aligned} E_{NTRU} &= P \cdot T_{NTRU}, \\ E_{RCPKC} &= P \cdot T_{RCPKC}. \end{aligned} \quad (62)$$

Since T_{NTRU} is greater than T_{RCPKC} by more than 27 times, then from (62):

$$\frac{E_{NTRU}}{E_{RCPKC}} = \frac{T_{NTRU}}{T_{RCPKC}} \geq 27. \quad (63)$$

From (63), RCPKC consumes twenty seven times less energy than NTRU using the same frequency.

Different frequencies: Since RCPKC is 27 times faster than NTRU, the former takes approximately the same run time on a 27 times lower clock frequency CPU than that of the latter. Dynamic and leakage power consumption, calculated for frequencies from [48] (p. 19) according to (21), are shown in Table 4.

Table 4. MSP430FR5969 microcontroller dynamic and leakage power consumption, P_{dyn} and P_{leak} , for frequencies from [48] (p. 19) at active supply voltages of 3 and 2.2 V.

Frequency (MHz)	2.2 V		3 V	
	P_{dyn} (μ W)	P_{leak} (nW)	P_{dyn} (μ W)	P_{leak} (nW)
1	48.4	44	90	60
16	774.4		1440	

It follows from Table 4 that $P_{leak} \ll P_{dyn}$, and it can be neglected. From Table 4, it follows that reducing the clock frequency from 16 to 1 MHz leads to a 16 times power consumption reduction from 1440 to 90 μ W. Note that MSP430FR5969, at a lower frequency, operates at a lower voltage: operating on a 1 MHz frequency at 2.2 V [48] (p. 19) results in 48.4 μ W of dynamic power consumption.

Hence, the total power reduction is $\frac{1440}{48.4} \approx 30$ times. Therefore, RCPKC, compared to NTRU, is better applicable to WSNs with power constrained devices.

9. Conclusions

In this paper, RCPKC is proposed, a secure and effective congruential, modulo q , public-key cryptosystem using big numbers. It uses the same encryption/decryption mechanism as NTRU does, but works with numbers. Contrary to NTRU, RCPKC is resistant to LBRA because its private key components, f and g , are chosen big with respect to \sqrt{q} to form a two-component vector with the norm exceeding Minkowski's boundary (27)–(30) for the shortest vector in a two-dimensional lattice and meeting (31). Hence, LBRA by the GLR algorithm returning the shortest vector in a two-dimensional lattice fails at finding the large norm private key vector, (f, g) .

In spite of the big numbers, f and r , meeting (36) used in RCPKC, it guarantees that the decryption correctness condition (15) holds (see (39)) due to the use of Conditions (33), (36), (38), (46) and (50) instead of Conditions (7), (11), and (12), used in the original insecure CPKC (see Sections 3.3–3.3.2) considered in [26]. It was found that the insecurity of the original CPKC stems from the use of Conditions (7), (11) and (12), defining smaller than \sqrt{q} numbers f, g, m, r meeting Minkowski's boundary (27) and the decryption correctness condition (15). RCPKC is resistant to the LBRA by GLR attack due to the special choice of the range for the random value, r , used in the encryption (13) that guarantees correctness condition (15) violation for the short vectors returned by GLR, but holding for the original private key, (f, g) . Section 6 shows also that the security of RCPKC with respect to other known attacks on NTRU is not less than that of NTRU, which allows us to conclude that RCPKC is more secure than NTRU. Section 7 proves the IND-CCA2 security of RCPKC asymmetric encryption padding (RAEP).

RCPKC uses numbers, i.e., minimal possible, degree zero, polynomials, which makes it about 27 times more effective in encryption and decryption than NTRU and more than three times more effective in encryption with respect to the fastest most recently published NTRU variant, BQTRU [21], as the experiments show (see Table 3). Compared to NTRU, RCPKC reduces the energy consumption at least 27 times, which allows increasing the life-time of unattended WSNs by more than 27 times.

As a future work, the proposed RCPKC will be applied to telemedicine to secure the data collected by medical sensors and cameras.

Supplementary Materials: The following are available online at <http://www.mdpi.com/1424-8220/20/16/4632/s1>: Code S1: Maple code of LBRA by GLR, Table S1: RCPKC performance experiments' results, Table S2: NTRU performance experiments' results, Example S1: Example of CPKC encryption/decryption, Example S2: LBRA attack using GLR against CPKC, Example S3: Example of RCPKC.1 encryption/decryption, Example S4: LBRA attack using GLR against RCPKC.1, Example S5: Example of finding RCPKC.2's random interval and LBRA by GLR failure.

Author Contributions: Conceptualization, A.I. and A.C.; funding acquisition, J.J.P.C.R.; investigation, A.I. and A.C.; methodology, A.I., A.C. and N.H.; software, J.J.P.C.R.; validation, A.A.-K.; visualization, Y.-A.D.; writing, review and editing, Y.-A.D., A.A.-K., and J.J.P.C.R. All authors read and agreed to the published version of the manuscript.

Acknowledgments: A.I., N.H., and Y.-A.D. would like to thank Palestine Technical University—Kadoorie for providing support. We thank the anonymous reviewers for their useful comments.

Funding: This research was partially funded by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under the Project UIDB/EEA/50008/2020; as well as by the Brazilian National Council for Research and Development (CNPq) via Grant No. 309335/2017-5. The APC was funded by Palestine Technical University—Kadoorie.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless Sensor Network Survey. *Comput. Netw.* **2008**, *52*, 2292–2330. [[CrossRef](#)]
2. Akyildiz, I.F.; Melodia, T.; Chowdhury, K.R. A survey on wireless multimedia sensor networks. *Comput. Netw.* **2007**, *51*, 921–960. [[CrossRef](#)]
3. Gungor, V.C.; Lu, B.; Hancke, G.P. Opportunities and Challenges of Wireless Sensor Networks in Smart Grid. *IEEE Trans. Ind. Electron.* **2010**, *57*, 3557–3564. [[CrossRef](#)]
4. Li, M.; Lou, W.; Ren, K. Data security and privacy in wireless body area networks. *IEEE Wirel. Commun.* **2010**, *17*, 51–58. [[CrossRef](#)]
5. He, D.; Kumar, N.; Chilamkurti, N.K. A secure temporal-credential-based mutual authentication and key agreement scheme with pseudo identity for wireless sensor networks. *Inf. Sci.* **2015**, *321*, 263–277. [[CrossRef](#)]
6. Feng, D.; Jiang, C.; Lim, G.; Cimini, L.J.; Feng, G.; Li, G.Y. A Survey of Energy-Efficient Wireless Communications. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 167–178. [[CrossRef](#)]
7. Howgrave-Graham, N.; Silverman, J.H.; Singer, A.; Whyte, W. NAEP: Provable Security in the Presence of Decryption Failures. *IACR Cryptol. ePrint Arch.* **2003**, *2003*, 172.
8. Howgrave-Graham, N.; Silverman, J.H.; Whyte, W. Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3. In *Cryptographers' Track at the RSA Conference*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 118–135.
9. Hermans, J.; Vercauteren, F.; Preneel, B. Speed Records for NTRU. In *Proceedings of the Topics in Cryptology—CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, 1–5 March 2010*; Pieprzyk, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 5985, pp. 73–88.
10. Kaur, I.; Kour, H.; Verma, A. Ticket based Secure Authentication Scheme using NTRU Cryptosystem in Wireless Sensor Network. *Int. J. Comput. Sci. Inf. Secur.* **2017**, *15*, 55.
11. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A Ring-Based Public Key Cryptosystem. In *Proceedings of the Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, OR, USA, 21–25 June 1998*; Buhler, J., Ed.; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1423, pp. 267–288.
12. Boorghany, A.; Sarmadi, S.B.; Jalili, R. On Constrained Implementation of Lattice-Based Cryptographic Primitives and Schemes on Smart Cards. *ACM Trans. Embed. Comput. Syst.* **2015**, *14*, 42:1–42:25. [[CrossRef](#)]
13. Behrens, G.; Weicht, J.; Schlender, K.; Fehring, F.; Dreimann, R.; Meese, M.; Hamelmann, F.; Thiel, C.; Försterling, T.; Wübbenhorst, M. Smart Monitoring System of Air Quality and Wall Humidity Accompanying an Energy Efficient Renovation Process of Apartment Buildings. In *From Science to Society*; Otjacques, B., Hitzelberger, P., Naumann, S., Wohlgemuth, V., Eds.; Springer: Cham, Switzerland, 2018; pp. 181–189.
14. Malla, A.; Sahu, P.; Mishra, M. A Novel Encryption Scheme for Secure SMS Communication. In *Advances in Electronics, Communication and Computing*; Kalam, A., Das, S., Sharma, K., Eds.; Springer: Singapore, 2018; pp. 467–478.
15. Jun, J.; Chen, H. A novel mutual authentication and key agreement protocol based on NTRU cryptography for wireless communications. *J. Zhejiang Univ. Sci.* **2005**, *6*, 399–404. [[CrossRef](#)]
16. Bailey, D.V.; Coffin, D.; Elbirt, A.J.; Silverman, J.H.; Woodbury, A.D. NTRU in Constrained Devices. In *Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2001, Third International Workshop, Paris, France, 14–16 May 2001*; Koç, Ç.K., Naccache, D., Paar, C., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2162; pp. 262–272.
17. Guillen, O.M.; Pöppelmann, T.; Mera, J.M.B.; Bongenaar, E.F.; Sigl, G.; Sepúlveda, J. Towards post-quantum security for IoT endpoints with NTRU. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, 27–31 March 2017*; pp. 698–703.
18. Kamal, A.A.; Youssef, A.M. Strengthening hardware implementations of NTRUEncrypt against fault analysis attacks. *J. Cryptogr. Eng.* **2013**, *3*, 227–240. [[CrossRef](#)]
19. Sedenion. Available online: <https://en.wikipedia.org/wiki/Sedenion> (accessed on 29 January 2020)
20. Alsaidi, N.M.; Yassein, H.R. BITRU: Binary Version of the NTRU Public Key Cryptosystem via Binary Algebra. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 1–6.
21. Bagheri, K.; Sadeghi, M.R.; Panario, D. A non-commutative cryptosystem based on quaternion algebras. *Des. Codes Cryptogr.* **2017**, *86*, 2345–2377. [[CrossRef](#)]

22. Banks, W.D.; Shparlinski, I.E. A Variant of NTRU with Non-invertible Polynomials. In Proceedings of the Cryptology—INDOCRYPT 2002, Third International Conference on Cryptology in India, Hyderabad, India, 16–18 December 2002; Menezes, A., Sarkar, P., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2551, pp. 62–70.
23. Coglianesi, M.; Goi, B. MaTRU: A New NTRU-Based Cryptosystem. In Proceedings of the Cryptology—INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, 10–12 December 2005; Volume 3797, pp. 232–243.
24. Gaborit, P.; Ohler, J.; Solé, P. CTRU, a Polynomial Analogue of NTRU. Technical Report, RR-4621, Inria. 2006. Available online: <https://hal.inria.fr/inria-00071964/document> (accessed on 14 August 2020)
25. Gaithuru, J.N.; Salleh, M. ITRU: NTRU-Based Cryptosystem Using Ring of Integers. *Int. J. Innov. Comput.* **2017**, *7*, 33–38.
26. Hoffstein, J.; Pipher, J.; Silverman, J.H. *An Introduction to Mathematical Cryptography*; Springer: New York, NY, USA, 2014; pp. 373–376.
27. Jarvis, K.; Nevins, M. ETRU: NTRU over the Eisenstein integers. *Des. Codes Cryptogr.* **2015**, *74*, 219–242, doi:10.1007/s10623-013-9850-3. [[CrossRef](#)]
28. Karbasi, A.H.; Atani, R.E. ILTRU: An NTRU-Like Public Key Cryptosystem Over Ideal Lattices. *IACR Cryptol. ePrint Arch.* **2015**, *2015*, 549.
29. Malekian, E.; Zakerolhosseini, A. OTRU: A non-associative and high speed public key cryptosystem. In Proceedings of the 2010 15th CSI International Symposium, Computer Architecture and Digital Systems (CADSD), Tehran, Iran, 23–24 September 2010; pp. 83–90.
30. Malekian, E.; Zakerolhosseini, A.; Mashatan, A. QTRU: quaternionic version of the NTRU public-key cryptosystems. *ISC Int. J. Inf. Secur.* **2011**, *3*, 29–42.
31. Thakur, K.; Tripathi, B. STRU: A Non Alternative and Multidimensional Public Key Cryptosystem. *Glob. J. Pure Appl. Math.* **2017**, *13*, 1447–1464.
32. Thakur, K.; Tripathi, B.P. BTRU, A Rational Polynomial Analogue of NTRU Cryptosystem. *Int. J. Comput. Appl.* **2016**, *145*, 22–24, doi:10.5120/ijca2016910769. [[CrossRef](#)]
33. Vats, N. NNTRU, a noncommutative analogue of NTRU. *arXiv* **2009**, arXiv:0902.1891.
34. Yassein, H.R.; Al-Saidi, N.M. HXDTRU Cryptosystem Based On Hexadecniion Algebra. In Proceedings of the 5th International Cryptology and Information Security Conference, Kota Kinabalu, Malaysia, 31 May–2 June 2016.
35. Yu, Y.; Xu, G.; Wang, X. Provably Secure NTRU Instances over Prime Cyclotomic Rings. In Proceedings of the Public-Key Cryptography—PKC 2017—20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, 28–31 March 2017; Volume 10174, pp. 409–434.
36. Stehlé, D.; Steinfeld, R. Making NTRUEncrypt and NTRUSign as Secure as Standard Worst-Case Problems over Ideal Lattices. *IACR Cryptol. ePrint Arch.* **2013**, *2013*, 4.
37. Seck, M.; Sow, D. BI-NTRU Encryption Schemes: Two New Secure Variants of NTRU. In *Algebra, Codes and Cryptology*; Gueye, C.T., Persichetti, E., Cayrel, P.L., Buchmann, J., Eds.; Springer International Publishing: Cham, Switzerland, 2019; pp. 216–235.
38. Wang, B.; Lei, H.; Hu, Y. D-NTRU: More efficient and average-case IND-CPA secure NTRU variant. *Inf. Sci.* **2018**, *438*, 15–31. [[CrossRef](#)]
39. Lenstra, A.K.; Lenstra, H.W.; Lovász, L. Factoring polynomials with rational coefficients. *Math. Ann.* **1982**, *261*, 515–534. [[CrossRef](#)]
40. Hoffstein, J.; Silverman, J.H.; Whyte, W. Estimated Breaking Times for NTRU Lattices. Technical Report #012, Version 2, NTRU Cryptosystems. 2003. Available online: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.6336&rep=rep1&type=pdf> (accessed on 14 August 2020)
41. IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. *IEEE Std 1363.1-2008* **2009**, 1–81, doi:10.1109/IEEESTD.2009.4800404. [[CrossRef](#)]
42. Ibrahim, A.; Chefranov, A.; Hamad, N. NTRU-Like Secure and Effective Congruential Public-Key Cryptosystem Using Big Numbers. In Proceedings of the 2019 2nd International Conference on New Trends in Computing Sciences (ICTCS), Amman, Jordan, 9–11 October 2019; pp. 20–26.
43. Silverman, J.H.; Whyte, W. Estimating Decryption Failure Probabilities for NTRUEncrypt. Technical Report #18, Version 1, NTRU Cryptosystems. 2003. Available online: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.3.7016&rep=rep1&type=pdf> (accessed on 14 August 2020).

44. Smith, C. *Environmental Physics*; Routledge: London, UK, 2002; pp. 1028–1029.
45. Beloglazov, A.; Buyya, R.; Lee, Y.C.; Zomaya, A.Y. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. *Adv. Comput.* **2011**, *82*, 47–111. [[CrossRef](#)]
46. Nikolic, B. Design in the Power-Limited Scaling Regime. *IEEE Trans. Electron Devices* **2008**, *55*, 71–83. [[CrossRef](#)]
47. Romli, N.; Minhad, K.; Reaz, M.; Amin, M.S. An overview of power dissipation and control techniques in cmos technology. *J. Eng. Sci. Technol.* **2015**, *10*, 364–382.
48. Texas Instruments Inc. *MSP430FR596x, MSP430FR594x Mixed-Signal Microcontrollers; SLAS704G*; 2018. Available online: <https://www.ti.com/lit/ds/symlink/msp430fr5969.pdf> (accessed on 14 August 2020).
49. Bourbaki, N. *Topological Vector Spaces: Chapters 1–5*, 1st ed.; Elements of Mathematics; Springer: Berlin/Heidelberg, Germany, 2003.
50. Maple. Available online: <https://www.maplesoft.com> (accessed on 29 January 2020).
51. Smeets, I.; Lenstra, A.; Lenstra, H.; Lovász, L.; Nguyen, P.Q.; Vallée, B. *The LLL Algorithm: Survey and Applications*, 1st ed.; Information Security and Cryptography; Springer: Berlin/Heidelberg, Germany, 2010.
52. Howgrave-Graham, N. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. In Proceedings of the Cryptology—CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2007; Menezes, A., Ed.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4622, pp. 150–169.
53. Hoffstein, J.; Pipher, J.; Schanck, J.M.; Silverman, J.H.; Whyte, W.; Zhang, Z. Choosing Parameters for NTRUEncrypt. In Proceedings of the Topics in Cryptology—CT-RSA 2017—The Cryptographers’ Track at the RSA Conference 2017, San Francisco, CA, USA, 14–17 February 2017; Handschuh, H., Ed.; Springer: Berlin/Heidelberg, Germany, 2017; Volume 10159, pp. 3–18.
54. Jaulmes, É.; Joux, A. A Chosen-Ciphertext Attack against NTRU. In Proceedings of the Cryptology—CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, CA, USA, 20–24 August 2000; Bellare, M., Ed.; Springer: Berlin/Heidelberg, Germany, 2000; Volume 1880, pp. 20–35.
55. Howgrave-Graham, N.; Nguyen, P.Q.; Pointcheval, D.; Proos, J.; Silverman, J.H.; Singer, A.; Whyte, W. The Impact of Decryption Failures on the Security of NTRU Encryption. In Proceedings of the Cryptology—CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2003; Boneh, D., Ed.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2729, pp. 226–246.
56. Gama, N.; Nguyen, P.Q. New Chosen-Ciphertext Attacks on NTRU. In Proceedings of the Public Key Cryptography—PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, 16–20 April 2007; Okamoto, T., Wang, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4450, pp. 89–106.
57. Howgrave-Graham, N.; Silverman, J.H.; Whyte, W. *A Meet-in-the-Middle Attack on an NTRU Private Key*; Technical Report #004, Version 2; NTRU Cryptosystems: June 2003. Available online: <https://assets.onboardsecurity.com/static/downloads/NTRU/resources/NTRUTech004v2.pdf> (accessed on 14 August 2020).
58. Kirchner, P.; Fouque, P. Revisiting Lattice Attacks on Overstretched NTRU Parameters. In Proceedings of the Cryptology—EUROCRYPT 2017—36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, 30 April–4 May 2017; pp. 3–26.
59. Whyte, W.; Etzel, M. Open Source NTRU Public Key Cryptography Algorithm and Reference Code. Available online: <https://github.com/{NTRU}OpenSourceProject/ntru-crypto> (accessed on 29 January 2020).
60. Victor, S. NTL: A Library for Doing Number Theory. Available online: <http://www.shoup.net/ntl/> (accessed on 29 January 2020).
61. Melhem, A.I. RCPKC Implementation using C. Available online: <https://github.com/anasnet/RCPKC-Project> (accessed on 5 August 2020).
62. Moore, D.S.; Kirkland, S. *The Basic Practice of Statistics*; WH Freeman: New York, NY, USA, 2007; Volume 2.

