Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

Article

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

Anas Ibrahim[1,2]*, Alexander Chefranov[2], Nagham Hamad[1], Yousef-Awwad Daraghmi[1], Ahmad Al-Khasawneh[3], Joel J. P. C. Rodrigues[4,5]

1 Palestine Technical University–Kadoorie, Tulkarm, Palestine; a.melhem@ptuk.edu.ps (A.I.); nagham.hamad@ptuk.edu.ps (N.H); y.awwad@ptuk.edu.ps (Y.D.)
2 Computer Engineering Department, Eastern Mediterranean University, North Cyprus; anas.ibrahim@emu.edu.tr (A.I.); alexander.chefranov@emu.edu.tr (A.C.)
3 Professor of Information Systems at Hashemite University, Jordan; akhasawneh@hu.edu.jo; (A.K.)
4 Federal University of Piauí, Teresina – PI, Brazil; joeljr@ieee.org (J.R.)
5 Instituto de Telecomunicações, 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 363 Cryptosystem Using Big Numbers. 2019 2nd International Conference on new Trends in Computing 364 Sciences (ICTCS) , Amman, Jordan, October 9-11, 2019, pp. 20–26..

Abstract: Wireless Sensor Networks (WSN) are the core of Internet of Things and require cryptographic protection due to the increase number of attacks. Cryptographic methods for WSN should be fast and consume low power as these networks consist of battery-powered devices and constrained microcontrollers. NTRU, the fastest and secure public key cryptosystem, uses high degree polynomials, and is susceptible to the lattice basis reduction attack (LBRA). CPKC, proposed by 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 integers 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 similar to 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 IND-CPA security is proved under the assumption of hardness of its one-way function. Due to the use of big numbers instead of high degree polynomials, RCPKC is about 24 (7) times faster in encryption (decryption) than NTRU. Also, RCPKC is more than three times faster than the most effective known NTRU variant, BQTRU. Compared to NTRU, RCPKC reduces energy consumption at least seven times that allows increasing life-time of unattended WSN more than seven times.

Keywords: Wireless sensor network; Internet of Things; random congruential public-key cryptosystem; lattice; NTRU; polynomial; lattice basis reduction attack; LLL algorithm; Gaussian lattice reduction; IND-CPA security

1. Introduction

Wireless Sensor Networks (WSN) play an important role in the development of Internet of Things (IoT). WSN consists 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], 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]. Also,energy saving is very important for WSN [6]. NTRU [7] is a Public Key Cryptosystem (PKC) standardized as IEEE P1363.1 and faster than RSA and ECC [8], and it is applicable cryptosystem for WSNs [9]. 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 [7]. These features of NTRU make it applicable to various applications, such as authentication for smart cards [10], encryption of user data in smart monitoring system [11], securing of SMS [12], mutual authentication and key agreement for wireless communications [13], embedded systems including microcontrollers and FPGAs [14], Internet of Things devices [15], and NTRU hardware implementation [16]. 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 only by the keys owner (receiver).

NTRU and its known variants [17–38], 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 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 running time in N 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, the NTRU uses polynomials with high degree N for the encryption/decryption mechanism [7,41], and this increases its time complexity. In addition, NTRU originally proposed in [7] also is not an IND-CPA cryptosystem as shown in [42]. Therefore, NTRU variants, shown in Section 2, try minimizing NTRU computational complexity by extending coefficients of the used polynomials, or using matrices of polynomials that allows preserving security level while decreasing the polynomial degree and to provide IND-CPA security. The extreme case is a polynomial of zero degree, that is a number, as used in CPKC, but CPKC as shown in [24] is insecure against LBRA by GLR. The authors of the NTRU applied its encryption/decryption mechanism in [24, p. 373-376] to integers modulo $q >> 1$, considering congruential public key cryptosystem (CPKC), and they found that CPKC is insecure since GLR reveals its private keys in about ten iterations. So, the CPKC is considered as a toy model of NTRU because "it provides the lowest dimensional introduction to the NTRU public key cryptosystem" [24, p. 374]. 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 previously proposed CPKC modification, RCPKC is originally proposed in [43] (we call it here RCPKC.1). However, our further studies of LBRA attack revealed some cases where the attack can succeed to compromise encrypted message in the originally proposed solution.

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 the Minkowski's boundary (29)-(32) 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 an IND-CPA secure cryptosystem more secure than NTRU because LBRA currently considered as one of the most effective attack against NTRU as well as a number of other attacks on NTRU are not applicable to RCPKC, whereas RCPKC resistance to other known attacks on NTRU is similar to that of NTRU. RCPKC is about 24 times faster in encryption and seven times faster in decryption than NTRU. Simplicity, speed, and security make RCPKC a good candidate cryptosystem for WSNs. The paper 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 [43].

- Hardness of RCPKC one-way (OW) function is proved.
- RCPKC semantic security, i.e. IND-CPA security, is proved under the assumption of the hardness of inverting an associated one-way function
- RCPKC performance is justified through implementation and comparison with the state-of-the-art ciphers.
- RCPKC better than NTRU applicability to WSNs is proved.

The rest of the paper is organized as follows. In Section 2, known NTRU variants are presented. In Section 3, overview of NTRU and CPKC is made, 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 CPKC private key/ message fails. In Section 5, RCPKC is presented. In Section 6, RCPKC security comparison versus NTRU is conducted. In Section 7, RCPKC OW function and IND-CPA security are considered. In Section 8, RCPKC performance comparison versus NTRU and its variants is presented, and 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, ring defining polynomial, or the polynomials are used as entries of such structures as matrices. NTRU variants overview follows.

NTRU variants differing in the choice of their coefficients. In [25], an NTRU variant, ETRU, is proposed working with polynomials over Eisenstein integer coefficients and faster than NTRU in encryption/decryption by 1.45/1.72 times. Karbasi and Atani [26] modified ETRU, called ILTRU [26], so that it works with irreducible cyclotomic polynomial over Eisenstein integer coefficients. NTRU variant, BITRU, working with polynomials over so called binary numbers, usually known as complex numbers is proposed in [18]. NTRU variant, QTRU, working with polynomials over hyper-complex four-component numbers, quaternions, is proposed in [28]. Also Bagheri and colleagues proposed NTRU variant, BQTRU, working over quaternions but with bivariate polynomials seven times faster than NTRU in encryption [19]. A variant of NTRU working with polynomials over 8-component hyper-complex numbers, octonions, called OTRU, is proposed in [27]. In [32], NTRU variant, HXDTRU, is proposed working with polynomials over 16-component hyper-complex numbers, hexadecnions, also known as sedenions [17]. Also, a variant of NTRU working with polynomials over 16-component hyper-complex numbers, sedenions, is proposed in [29]. A variant of NTRU, called CTRU, working with polynomials coefficients of which are also polynomials in one variable over a binary field is proposed in [22]. Also, a variant of NTRU working with polynomials coefficients of which are polynomials in one variable over rational field called BTRU is proposed in [30].

NTRU variants working with different rings. NTRU variant that works with polynomials with prime cyclotomic rings is proposed in [33]. A variant of NTRU working with non-invertible polynomials is proposed in [20].

NTRU variants working with polynomials inside more complicated structures. NTRU variant working with square matrices of polynomials is proposed in [21] and showed 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 is proposed in [31]. Apart from the polynomial variants, an NTRU-like cipher over the ring of integers, called ITRU, is proposed in [23] without referencing to CPKC [24]. In ITRU [23], Table 1, p. 34, the NTRU model specified above is given but a model for the proposed ITRU is not defined. Its Algorithm 1, [23], p. 35, describes the keys generation, and, hence, it shall be made by the keys owner (receiver). On the other hand, in the Section IV. A, Parameter selection, [23], p. 37, the most important parameter, q, is selected by the sender (which encrypts a message using the public key, $h' = 423642$ and random value, $r' = 19$, in [23], (19), p. 37) with the help of the private keys, f', g', that contradicts to the NTRU model: the secret key is known to

the keys owner only that uses for decryption the private key only, whereas the public key is used for encryption by the public user only.

NTRU variants provide IND-CPA security. NTRU [7] is not IND-CPA cryptosystem as shown in [42] and presented in Section 7.3. Various NTRU variants have been proposed to resolve this issue, Stehlé and Steinfeld proposed IND-CPA NTRU variant by changing NTRU ring to $(x^N + 1)$ with N a power of 2, and adding small error from LWE distribution [34]. Seck and Sow in [35] also provided two variants using the ring $(x^N + 1)$ and the assumption of the hardness of Ring Learning With Error (Ring-LWE) problem. Howgrave-Graham et al. proposed IND-CPA secure variant called NAEP [36,37] using message padding and hashing. Wang et al. in [38] proposed IND-CPA variant, D-NTRU, using two-step encryption.

In conclusion, the NTRU variants try minimizing NTRU computational complexity by extending coefficients of the used polynomials, or using matrices of polynomials that allows preserving 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. Also, the ITRU can be used by the keys owner only for encryption and decryption messages, but cannot be used by a public user knowing the public key only, and, hence, it is not compatible with NTRU model of use.

3. Preliminaries

3.1. Overview of NTRU

NTRU [24] uses the rings

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

elements of which are polynomials modulo $x^N - 1$ with coefficients in $\mathbf{Z}_q, \mathbf{Z}_p$ respectively, where $p < q$, p, N are primes, $\gcd(p, q) = \gcd(N, q) = 1$.
Let $T(d_1, d_2)$ be a subset of R_q with polynomials having d_1 coefficients equal to 1, d_2 coefficients equal to -1, and the rest coefficients equal to zero.
The secret polynomials, f, g, are of the form

$$f \in T(d + 1, d), \quad g \in T(d, d). \tag{1}$$

The public polynomial, h, is computed as follows:

$$h = F_q \cdot g \bmod q, \tag{2}$$

where F_q is the inverse of f modulo q. A random polynomial, r, and message, m, are of the form:

$$r \in T(d, d), \quad m \in R_p. \tag{3}$$

NTRU encryption is as follows:

$$e = p \cdot r \cdot h + m \bmod q. \tag{4}$$

NTRU decryption consists of two steps:

Step 1: The private key, f, is applied to (4):

$$\begin{aligned} a &= f \cdot e \bmod q \\ &= p \cdot r \cdot g + f \cdot m. \end{aligned} \tag{5}$$

Step 2: The inverse of f modulo p is applied to (5) after the polynomial a is center-lifted (making coefficients of a by absolute value less than $q/2$).

3.2. Overview of CPKC

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

$$f < \sqrt{q/2}, \quad \sqrt{q/4} < g < \sqrt{q/2}, \tag{6}$$

$$\gcd(f, qg) = 1, \tag{7}$$

where q is a public integer.
The first secret value, f, has inverses modulo g and q, that is, F_g, F_q, respectively, by virtue of (7):

$$1 = f \cdot F_g \bmod g, \quad 1 = f \cdot F_q \bmod q. \tag{8}$$

A public value, h, is computed using (6), (8) as follows

$$h = F_q \cdot g \bmod q. \tag{9}$$

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:

$$0 < m < \sqrt{q/4}. \tag{10}$$

A random integer, r, is chosen as follows:

$$0 < r < \sqrt{q/2}. \tag{11}$$

3.2.1. CPKC Encryption

The ciphertext, e, is computed using (9)-(11) as follows:

$$e = r \cdot h + m \bmod q. \tag{12}$$

3.2.2. CPKC Decryption

Decryption is described by Steps 1, 2 below:
Step 1: Multiply the ciphertext (12) by f getting

$$a = f \cdot e \bmod q$$

$$= r \cdot f \cdot F_q \cdot g + f \cdot m \bmod q. \tag{13}$$

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

$$|r \cdot g + f \cdot m| < q, \tag{14}$$

where (8), (9), and (12) are used. CPKC decryption correctness condition (14) holds under conditions (6), (10), (11):

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

Thus, the parameters, f, g, r, are selected small compared to q (see (6), (10), (11)) to meet the CPKC correctness decryption condition (14) used in Step 2 of the decryption.
Step 2: Multiply (13) by F_g, getting

$$m = a \cdot F_g \bmod g, \tag{15}$$

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

where (8) is used and the contributor with the factor g in (13) vanishes due to (14).

3.2.3. Example of CPKC Encryption/ Decryption

Example 1. *CPKC Encryption/ Decryption.*

The example is close to Example 7.1, from [24, p. 375]. Let according to (6), (7), (10), q = 122430513839, f = 231233, g =195696, and m =12345.
According to (8), F_g=127505, and F_q=54368439252. Public key component, h, is calculated by (9):

$$h = F_q \cdot g \bmod q = 107143708775.$$

Let according to (11), r =10101. The ciphertext, e, is computed according to (12):

$$e = r \cdot h + m \bmod q = 95290525699. \tag{16}$$

To decrypt the ciphertext (16), apply **Step 1**, equation (13):

$$a = f \cdot e \bmod q = r \cdot g + f \cdot m = 4831296681. \tag{17}$$

In **Step 2**, the message m is retrieved using (15):

$$m = F_g \cdot a \bmod g = 12345. \tag{18}$$

Thus, in (18), the plaintext, mm is revealed. It can be seen that CPKC encryption/decryption procedure (12), (13), (15), works correctly due to (14) holding. Note that the norm of the vector, $(f,g) = \sqrt{f^2+g^2} = 302928.4$ is small compared to $\sqrt{q} = 759250123.0$.

3.3. 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 :

$$P = V \cdot I, \tag{19}$$

$$E = P \cdot T, \tag{20}$$

where V is the potential difference measured in volts (V), and I is the electric current measured in amperes (A), 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]:

$$P_{cpu} = P_{dyn} + P_{sc} + P_{leak}. \tag{21}$$

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

$$P_{leak} = V \cdot I_{leak}, \tag{22}$$

where V is the supply voltage, and I_{leak} is 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]:

$$P_{dyn} = \boldsymbol{a} \cdot C \cdot V^2 \cdot f, \tag{23}$$

where $\boldsymbol{a}$ 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 $\boldsymbol{a}$ = 0.5 [47]. MSP430FR5969, a Texas Instruments microcontroller with capacitance $C = 20pF$ [48, Table 5-12], active supply voltage

from 1.8, .., 3.6 V [48, p. 1], clock frequency from 1, .., 16 MHz [48, p. 18], 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 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 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, $||x||$, $(x \cdot y)$, $\lfloor a \rceil$, and R, denote Euclidean norm [49] of the vector x, dot product of the vectors, x and y, rounding of the real number, a, and the set of real numbers, respectively. Let $E(V_1, V_2) \subset R^2$ be a 2-dimensional lattice with basis vectors, V_1 and V_2:

$$E(V_1, V_2) = \{a_1 V_1 + a_2 V_2 : a_1, a_2 \in \mathbf{Z}\}. \tag{24}$$

GLR algorithm [24, p. 437], shown in Code 1, on 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 $m = \lfloor (v_1 \cdot v_2)/||v_1||^2 \rceil$.
  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.

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 (9), it can be noticed that for any pair of integers, F and G, satisfying:

$$G = Fh \bmod q, \quad F = \mathcal{O}(\sqrt{q}), \quad G = \mathcal{O}(\sqrt{q}), \tag{25}$$

(F, G) is likely to serve as the first two components, f, g, of the private key, SK [24, p. 376]. Equation (25) can be written as $F \cdot h + q \cdot n = G$, where n is an integer. So, our task is to find a pair of comparatively small by absolute value integers, (F, G), such that

$$F \cdot V_1 + n \cdot V_2 = (F, G), \tag{26}$$

where $V_1 = (1, h)$ and $V_2 = (0, q)$ are basis vectors, at least one of them having Euclidean norm of order $\mathcal{O}(q)$. Similarly, CPKC message recovery problem can be formulated as SVP in the two-dimensional lattice, $E(V_1, V_2)$, where V_1 ,V_2 are from (26). It can be also noticed from (12), that for any pair of integers, (RR, EM), satisfying:

$$EM = RR \cdot h \bmod q, \quad RR = \mathcal{O}(\sqrt{q}), \quad EM = \mathcal{O}(\sqrt{q}), \tag{27}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

(RR, EM) is likely to serve as the vector $(r, e-m)$ since the encryption equation (12) can be written as $r \cdot h + q \cdot n = e - m$, where n is an integer. So, our task is to find a pair of comparatively small by absolute value integers, (RR, RM), such that

$$RR \cdot V_1 + n \cdot V_2 = (RR, EM). \tag{28}$$

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, r are of the order of $\mathcal{O}(\sqrt{q})$. Comparing (26) and (28), 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 2 is our implementation using Maple [50] of the LBRA by GLR based on Code 1.

Code 2. *Maple code of LBRA by GLR on CPKC private key/message returned as the shortest vector w=v1 of the lattice E(V1, V2), where V1, V2 are from (26).*

```
1  GLR := proc(f, g, h, q)
2  local   fg, fgnorm, v1norm, v2norm, a, tmp, i, μ := 10;
3  global result, counter, flag, v1, v2;
4  flag := 0;
5  v1 := Vector[column]([1, h]);
6  v2 := Vector[column]([0, q]);
7  fg := Vector[column]([f, g]);
8  result := Matrix(2, 1);
9  fgnorm := evalf(norm(fg, 2));
10 a := 1 :
11 counter := 0 :
12 while a ≠ 0 do
13    v1norm := evalf(norm(v1, 2));
14    v2norm := evalf(norm(v2, 2));
15    if v2norm < v1norm then
16       tmp := v1;
17       v1 := v2;
18       v2 := tmp;
19    end if;
20    if f=v1[1] or f=v2[1] then
21       if g=v1[2] or g=v2[2] then
22          flag := 1;
23          print("keys are found");
24       end if;
25    end if;
26       if v1norm ≤ μ · fgnorm then
27          result := ⟨result|v1⟩;
28       end if;
29       if v2norm ≤ μ · fgnorm then
30          result := ⟨result|v2⟩;
31       end if;
32       a := round(DotProduct(v1, v2) / (norm(v1, 2)^2));
33       v2 := v2 − a·v1;
34       counter := counter + 1 :
35 end do;
36 print(flag);
37 result := abs(result) :
38 result := DeleteColumn(result, [1]);
39 end proc:
```

Note that lines $20-30$ in Code 2 are added to support the proposal of RCPKC in Section 5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

LBRA by GLR using Code 2 on CPKC private key/message for the data from the Example 1, finds in 9 iterations the shortest vector, $v1$ = (231233, 195696) as shown in Figure 1. The shortest vector, $v1$, found by GLR corresponds to the private key components, (f, g), because they were selected small, having order $\mathcal{O}(\sqrt{q})$ values according to (6). The message related vector, $(r, e-m)$, is not disclosed in the attack because e= $\mathcal{O}(q)$ in the Example 1.

$f := 231233;$

$$231233 \quad (1)$$

$g := 195696;$

$$195696 \quad (2)$$

$q := 122430513839;$

$$122430513839 \quad (3)$$

$m := 12345;$

$$12345 \quad (4)$$

$r := 10101;$

$$10101 \quad (5)$$

$igcdex(f, q, 'Fq', 'qz') :$
$h := Fq \cdot g \bmod q$

$$107143708775 \quad (6)$$

$e := r \cdot h + m \bmod q;$

$$95290525699 \quad (7)$$

$GLR(f, g, h, q) :$

"keys are found"

$$1 \quad (8)$$

$v1$

$$\begin{bmatrix} 231233 \\ 195696 \end{bmatrix} \quad (9)$$

$counter$

$$9 \quad (10)$$

Figure 1. Screenshot of LBRA by GLR using Maple Code 2 on CPKC for the data from the Example 1 finding the private key components,(f, g)=v1, in 9 iterations.

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 CPKC Private Key/ Message

LBRA by GLR succeeds in finding CPKC private key since it, by using settings (6), 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 2-dimensional lattice:

$$\lambda \leq \sqrt{\lambda_2} Vol(L)^{1/2}, \quad (29)$$

where λ_2=$2/\sqrt{3} \approx 1.154$ is the Hermite's constant [51, p. 41], and $Vol(L)$ is the volume of the lattice which is equal to q for the lattice $L = E(V_1, V_2)$ where V_1, V_2 are defined in (26). Therefore, (29) can be written as follows:

$$\lambda \leq \alpha \sqrt{q}, \quad (30)$$

where α=$\sqrt{\lambda_2} \approx 1.07$. From (30), one gets for the relative norm,

$$\lambda' = \frac{\lambda}{\sqrt{q}}, \quad (31)$$

the following inequality (32):

$$\lambda' \leq \alpha. \tag{32}$$

GLR fails attacking CPKC private key/message when (32) is not satisfied for the secret vector relative norm (f,g), i.e. if

$$||(f,g)||/\sqrt{q} > \alpha \tag{33}$$

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

CPKC selects small values for private key (f,g) in (6) to satisfy decryption correctness condition (14). Hence, our goal is to propose in Section 5 a modification for CPKC, that is RCPKC, where (f,g) satisfies (33) and provide 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 3.2, so that it becomes resistant to GLR attack, is proposed.

5.1. RCPKC Main Ideas

The main two ideas of RCPKC are:

- Contrary to the settings (6) of CPKC, which uses secret key (f,g) with 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 (26), RCPKC [43] (we call it in this section RCPKC.1) is originally proposed having private key (f,g) with a large norm meeting (33) so that it cannot be returned by LBRA using GLR as the SV but (f,g) also meets (14) due to the skew in its components.
- However, as mentioned in Section 4.1 that for any pair of integers, F and G, satisfying (25), (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 decryption correctness condition (14), and thus may be used for the correct plaintext message disclosure as shown in Example 3. That is why, RCPKC.1, Section 5.2, before encrypting by (12), (contrary to CPKC using a random number from the predefined range (11)), 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 (26)), so that decryption correctness condition (14) holds for (f,g) but does not hold for (F,G) that leads to the failure of LBRA using GLR on RCPKC.1. Such interval defined in (42) - (44) for RCPKC.1 found to be vulnerable to GLR attack. Therefore, an enhanced RCPKC proposed herein (we call it in this section RCPKC.2) with more tight interval for r is defined in (48), (52), and (53), so that such attack is inactive.

Thus, RCPKC.2 assumes that the private key owner selects a range for a random value, r (used in encryption (12)), based on the secret key, (f,g), and respective SV, (F,G), in the lattice, $E(V_1,V_2)$, defined by (26), 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 which might happen that the range for random numbers such kind defined may be rather narrow and, thus, security of RCPKC may suffer. But as it is going to be shown the range is rather large and may significantly exceed the range for a secret message.

In Subsection 5.2, CPKC is modified to RCPKC.1 [43] so that the secret key, (f,g), meets (14) and (33). In Subsection 5.3, we consider Example 3, of GLR attack on RCPKC.1. In Subsection 5.4, RCPKC.1 is further modified to RCPKC.2, so that it becomes immune against LBRA attack. In Subsection 5.5, Example 4 shows GLR attack failure to disclose RCPKC.2 encrypted message.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

5.2. RCPKC.1 [43] Description

To meet (33), it is required that

$$f, r \geq \alpha \cdot \sqrt{q}. \tag{34}$$

The LBRA by GLR failure condition (33) holds if (34) is true since

$$\frac{||(f,g)||}{\sqrt{q}} = \frac{\sqrt{f^2+g^2}}{\sqrt{q}} = \frac{\sqrt{\alpha^2 \cdot q + g^2}}{\sqrt{q}} > \alpha,$$

$$\frac{||(r,e-m)||}{\sqrt{q}} = \frac{\sqrt{r^2+(e-m)^2}}{\sqrt{q}} = \frac{\sqrt{\alpha^2 \cdot q + (e-m)^2}}{\sqrt{q}} > \alpha,$$

for $g, e-m \neq 0$. Condition (34), in RCPKC.1, substitutes for the conditions (6), (11) on f, r, in CPKC. The message, m, and the private key, g, instead of (10), (6), used in CPKC, are redefined in RCPKC.1 as follows:

$$2^{mgLen} > g \geq 2^{mgLen-1} > m \geq 0, \tag{35}$$

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

For RCPKC.1, correctness decryption condition (14) shall hold, that is true (see (41)) when f, r values in addition to (34) meet (36):

$$\frac{q}{2 \cdot 2^{mgLen}} > f, r. \tag{36}$$

Since

$$q = 2^{qLen}. \tag{37}$$

Then, (34) and (36) can be rewritten:

$$2^{qLen-mgLen-1} > f, r \geq \alpha \cdot 2^{qLen/2}. \tag{38}$$

To have a non-empty range for f, r, of the width at least $\alpha \cdot 2^{qLen/2}$, from (38), the following condition is obtained:

$$\frac{2^{qLen/2}}{2 \cdot \alpha} > 2^{mgLen+1}. \tag{39}$$

By defining $\beta = \log_2 1/(2 \cdot \alpha) \approx -1.103$, (39) shows that

$$2^{\beta} \cdot 2^{qLen/2} > 2^{mgLen+1},$$

$$qLen + 2 \cdot \beta > 2 \cdot (mgLen + 1),$$

$$qLen > 2 \cdot (mgLen + 1 - \beta). \tag{40}$$

Let's show that the decryption correctness condition (14) holds when (35), (38), and (40) hold:

$$r \cdot g + f \cdot m < 2^{qLen-mgLen-1} \cdot 2^{mgLen} + 2^{qLen-mgLen-1} \cdot 2^{mgLen-1}$$

$$< 2^{qLen-1} + 2^{qLen-1} = 2^{qLen} = q. \tag{41}$$

Thus, for RCPKC.1, norm of (f, g) meets (33), and decryption correctness condition (41) holds. We need additionally that decryption correctness condition (41) is violated for (F, G), that is the SV obtained in the result of GLR attack on the lattice $E(V_1, V_2)$ defined by (26). Hence, it cannot be used as a private key for the plaintext message correct decryption.

Inequality (38) defines a range for r so that f, g, r, m meet (14). Now, we define constant on r,

$$r \geq rmin \geq (q + g|F|)/|G| \tag{42}$$

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

such that F, G, r, m violate (14). Using (42) and (35):

$$\begin{aligned} |G \cdot r + F \cdot m| &\geq |G| \cdot |r| - |F| \cdot m \geq \frac{|G|(q + g|F|)}{|G|} - |F| \cdot m \\ &\geq q + g|F| - m|F| > q. \end{aligned} \tag{43}$$

Thus, inequality (38) is used for f, but for r from (42) and (38), we have

$$2^{qLen-mgLen-1} > r \geq max(\alpha \cdot 2^{qLen/2}, rmin). \tag{44}$$

For RCPKC security, range defined by (44) shall be rather large, $\max(\alpha \cdot 2^{qLen/2}, rmin)$, hence:

$$2^{qLen-mgLen-1} \geq 2 \cdot max(\alpha \cdot 2^{qLen/2}, rmin). \tag{45}$$

Thus, RCPKC.1 is defined as follows.

RCPKC.1 Definition:
The private key components, (f, g), meet (7), (8), (35), and (36), where $qLen$, $mgLen$ meet (40) and (45), where (F, G) is an SV obtained in the result of GLR attack on the lattice $E(V1, V2)$ defined by (26). The public key component, h, is defined by (9). Message, m, meets (35), and random integer, r, is selected from the range defined in (42), (44). Encryption and decryption follow (12), and (13), (15), respectively (see Sections 3.2.1, and 3.2.2). Decryption correctness condition (14) is proved for RCPKC.1 in (41).

5.3. Examples of RCPKC.1 Encryption/Decryption and LBRA Attack Against RCPKC.1

Let us consider now Example 2 showing RCPKC.1 encryption and decryption processes.

Example 2. *RCPKC.1 Encryption/Decryption.*

For calculations, Maple is used.

Let $mgLen = 16$, $qLen = 80$, meeting (40), $q = 2^{qLen}$, private key components, private key component, $g = 2^{16} - 1$, is selected to meet (35). On the other hand, private key component, $f = 1,351,417,702,001$, is selected to meet (38). See (1)-(6) in Figure 2. According to (8), F_q, and F_g are calculated in (8) of Figure 2. Then, public key component, h, is computed using (9) as shown in (9) of Figure 2.

To compute the interval from which r is selected (44) and (45), lower boundary is found in (10)-(15) in Figure 2.

To encrypt message $m = 14$, random number $r = 1,176,477,442,250$ is selected from the interval. Then, ciphertext e, is calculated using (12) in (18) of Figure 2. For decryption, in the first step, according to (13), we find a by as the product of the ciphertext, e, and the private key f modulo q as shown in (19) of Figure 2. In the second decryption step, according to (15), we multiply, a, by F_g to get the message m as we can see in (20) of Figure 2. Hence, the message, m, is correctly retrieved.

Example 3. *LBRA attack using GLR against RCPKC.1 in settings of Example 2.*

Now, we try attacking RCPKC.1 in Example 2, using GLR Code 2. GLR terminates in 18 iterations finding $v1 = (F, G) = (-459459339518, -894561206306)$ that is neither (f, g) nor $(r, e - m)$ as shown in (21)-(25) Figure 3. Since $\gcd(F, q) = \gcd(F, G) = 2$, F has no inverses modulo q and G, and $v1$ cannot be used to decrypt the ciphertext.

Let's try the second shortest vector $v2 = (-207496671842665114072133, 229534132287)$ that is neither (f, g) nor $(r, e - m)$ as shown in (26)-(30) Figure 3. When using (F, G) for decryption of e, we get, $m1 = 65549 \neq m = 14$ as shown in (31)-(33) Figure 3. Thus, actually, ciphertext decryption fails if using any of the shortest vectors returned by GLR.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

Settings of RCPKC−1

$mgLen := 16$:
$m := 14$:
$qLen := 80$:
$q := 2^{qLen}$;

$$1208925819614629174706176 \quad (1)$$

$g := 2^{16} - 1$:
$is(g < 2^{mgLen}$ **and** $g \geq 2^{mgLen-1})$;

$$true \quad (2)$$

Lower boundary for f

alpha := 1.07 :
$lb :=$ ceil(alpha·sqrt(q)) ;

$$1176477442000 \quad (3)$$

Upper boundary for f, r

$Ub := 2^{qLen - mgLen - 1}$;

$$9223372036854775808 \quad (4)$$

$f := 1351417702001$;

$$1351417702001 \quad (5)$$

$is(f < Ub$ **and** $f \geq lb)$;

$$true \quad (6)$$

$gcd(f, q \cdot g)$;

$$1 \quad (7)$$

Computing inverse of f **modulo** q, g

igcdex(f, q,'Fq','qz') :
igcdex(f, g,'Fg','gz') :
Fq;
Fg;

$$154260404770580979079825$$
$$2291 \quad (8)$$

Computing public key h

$h := Fq \cdot g$ **mod** q;

$$417923022495305103287663 \quad (9)$$

Applying GLR to set lower boundary for r

$GLR(f, g, h, q)$

$$\begin{bmatrix} 1133958117 & 459459339518 & 26960316553 & 459459339518 & 891958362483 \\ 44728059201205 & 894561206306 & 2683683553383 & 894561206306 & 894561140771 \end{bmatrix} \quad (10)$$

$v1$

$$\begin{bmatrix} -459459339518 \\ -894561206306 \end{bmatrix} \quad (11)$$

$F :=$ abs($v1$[1])

$$459459339518 \quad (12)$$

$G :=$ abs($v1$[2])

$$894561206306 \quad (13)$$

$$rmin := \frac{(q + g \cdot F)}{G}$$

$$1351417702001 \quad (14)$$

max($rmin$, lb)

$$1351417702001 \quad (15)$$

$is(Ub > 2 \cdot rmin)$

$$true \quad (16)$$

$r := lb + 250$;

$$1176477442250 \quad (17)$$

Encrypt with RCPKC−1

$e := h \cdot r + m$ **mod** q

$$128263397495019445250468 \quad (18)$$

Decrypt with RCPKC−1

$a := f \cdot e$ **mod** q

$$77119369025681764 \quad (19)$$

$m0 := a \cdot Fg$ **mod** g

$$14 \quad (20)$$

Figure 2. Screenshot of Example 2 Maple code for RCPKC.1 encryption/decryption.

$GLR(f, g, h, q);$

$$\begin{bmatrix} 1133958117 & 459459339518 & 26960316553 & 459459339518 & 891958362483 \\ 44728059201205 & 894561206306 & 2683683553383 & 894561206306 & 894561140771 \end{bmatrix} \quad (21)$$

$counter$

$$18 \quad (22)$$

$v1$

$$\begin{bmatrix} -459459339518 \\ -894561206306 \end{bmatrix} \quad (23)$$

$F := v1[1]$

$$-459459339518 \quad (24)$$

$G := v1[2]$

$$-894561206306 \quad (25)$$

$gcd(F, q);$
$gcd(F, G);$

$$2$$
$$2 \quad (26)$$

$v2$

$$\begin{bmatrix} 891958362483 \\ -894561140771 \end{bmatrix} \quad (27)$$

$F := v2[1]$

$$891958362483 \quad (28)$$

$G := v2[2]$

$$-894561140771 \quad (29)$$

$gcd(F, G \cdot q)$

$$1 \quad (30)$$

$igcdex(F, q, 'Finvq', 'qz') :$
$igcdex(F, G, 'FinvG', 'Gz') :$
$Finvq;$
$FinvG;$

$$-207496671842665114072133$$
$$229534132287 \quad (31)$$

$A := F \cdot e \bmod q$

$$15649481679660831880 6188 \quad (32)$$

$m1 := FinvG \cdot A \bmod G$

$$65549 \quad (33)$$

$is(|r \cdot G + F \cdot m| < q)$

$$true \quad (34)$$

$A := F \cdot e \bmod q$

$$15649481679660831880 6188 \quad (35)$$

$A := A - q$

$$-10524310028180208558999 88 \quad (36)$$

$m2 := A \cdot FinvG \bmod G$

$$14 \quad (37)$$

Figure 3. GLR attack against RCPKC.1 in settings of Example 2

It has been noticed by anonymous reviewer that the value of $F \cdot e = r \cdot G + F \cdot m$ could be negative, but still satisfy correctness decryption condition in the absolute value

$$|r \cdot G + F \cdot m| < q.$$

Therefore, we can see that

$$F \cdot e = r \cdot G + F \cdot m = 156494785800294925503676 = -105243103381433424920250 0 \bmod q,$$

and $|-105243103381433424920250 0| < q = 120892581961462917470617 6$ as shown in (34) Figure 3. Thus, decryption correctness condition (41) holds. On the other hand, $A = (F \cdot e \bmod q) - q = -105243103381433424920250 0$. And the plaintext is restored as $m3 = (F^{-1} \bmod G) \cdot A \bmod G = -509 = 14 \bmod 523$, that is equal to $m = 14$ as shown in (35)-(37) Figure 3. Thus, the GLR attack succeeds revealing the plaintext message in the conditions of Example 2. Herein, it is necessary to be noticed that RCPKC.1 can be attacked by any of the short vectors returned by GLR.

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

5.4. RCPKC.2 Proposal

In order to resist GLR attack shown in Example 3, the definition of the region from which r is selected, should consider all SVs with norm less than a threshold $\mu||(f,g)||$ as follows.

Random interval defined in (42), (44), and (45) using only the SV obtained by GLR attack on the lattice $E(V_1, V_2)$ defined by (26), must be modified to include all the SVs with norm less than norm of secret key, by threshold $\mu|(f,g)|$. Hence, all vectors (F_i, G_i) obtained in the course of GLR reduction that have norms

$$||(F_i, G_i)|| < \mu||(f,g)||, i = 1, .., N, \tag{46}$$

where N is the number of (F, G) pairs satisfying (46), μ is a threshold, e.g., $\mu = 10$, and then it must be checked that

$$(\forall i = 1, .., N)((F_i, G_i) \neq (f,g)). \tag{47}$$

If (47) is violated, i.e. one of the vectors in the list is our vector (f,g), then another (f,g) is used.

Inequality (38) defines a range for r so that f, g, r, m meet (14). Now, constraint on r is defined as follows:

$$q/g - f \geq rmax \geq r \geq rmin \geq (q + g \cdot \max_{i=1,..,N} |F_i|) / \min_{i=1,..,N} |G_i|, \tag{48}$$

such that F_j, G_j, r, m violate (14) for any $j = 1, .., N$. We require also that

$$h \cdot rmin > q. \tag{49}$$

Using (48) and (35), it is noticed that actually decryption correctness condition (14) for any $j = 1, .., N$, is violated:

$$\begin{aligned} |G_j \cdot r + F_j \cdot m| &\geq |G_j \cdot r| - |F_j \cdot m| \geq |G_j| \cdot \frac{q + g \cdot \max_{i=1,..,N} |F_i|}{\min_{i=1,..,N} |G_i|} - |F_j \cdot m| \\ &\geq q + g \cdot \max_{i=1,..,N} |F_i| - |F_j \cdot m| > q. \end{aligned} \tag{50}$$

From (35), (48), it is also perceived that the decryption correctness condition (14) holds for the original (f,g):

$$g \cdot rmax + f \cdot m \leq g(q/g - f) + f \cdot m = q - f \cdot g + f \cdot m < q \tag{51}$$

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

Thus, inequality (38) is used for f, but for r from (48) and (38):

$$rmax > r \geq max(\alpha \cdot 2^{qlen/2}, rmin). \tag{52}$$

For RCPKC.2 security, range defined by (52) shall be rather large, such as, e.g., max$(\alpha \cdot 2^{qlen/2}, rmin)$, hence, it is desirable having:

$$rmax \geq 2 \cdot max(\alpha \cdot 2^{qlen/2}, rmin). \tag{53}$$

In order to provide IND-CPA security (see Section 7.3), it is required to have

$$\gcd(g, q) > 1. \tag{54}$$

Thus, RCPKC.2 proposal follows.

RCPKC.2 Proposal:

The private key components, (f, g), meet (7), (8), (35), (36), and (54), where $qLen$, $mgLen$ meet (40) and (45) The public key component, h, is defined by (9). Message, m, meets (35), and random integer, r, is selected from the range defined in (48), (49), and (52). Encryption and decryption follow (12), and (13), (15), respectively (see Sections 3.2.1, and 3.2.2). Decryption correctness condition (14) is proved for RCPKC in (51).

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 the both employ GLR and follow the same encryption/decryption procedures.

5.5. Example of RCPKC.2 Encryption/Decryption and LBRA by GLR Failure

This example aims to show the encryption/decryption process of RCPKC.2, and how LBRA using GLR fails to compromise RCPKC.2 private key/message.

Example 4. *RCPKC.2 encryption and decryption, and GLR failure to find RCPKC.2 secret key/message.*

For calculations, Maple is used.

Let $mgLen = 16$, $qLen = 80$, meeting (40), $q = 2^{qLen}$, private key components, $g = 65,535$, and $f = 1,351,417,702,001$, are selected to meet (35) and (38) respectively as shown in (2) and (3) of Figure 4.

Values F_q and F_g are found in (4) and (5) of Figure 4. The public key component, h, is calculated according to (9) as shown in (6) of Figure 4. To select random r, GLR algorithm shown in Code 2 is launched with inputs $V_1 = (1, h)$ and $V_2 = (0, q)$. GLR terminates in 18 iterations as shown in (8) of Figure 4, with 5 pairs (F_i, G_i) satisfying (46) shown in (7) of Figure 4, it is noticed that none of these vectors is equal to (f, g). Hence, (47) is satisfied. Maximum F_i and minimum G_i are found in (9), and (10) of Figure 4; value $rmin$ is defined according to (48) as shown in (11), $rmin$ also satisfies (49) as shown in (18) of Figure 4. $rmax$ is calculated in (12) of Figure 4. After calculating $max(\alpha \cdot 2^{qlen/2}, rmin)$ in (13) of Figure 4, it is perceived that (53) is satisfied as shown in (14) of Figure 4. Thus, r is selected from (52) as shown in (15) of Figure 4. For the message $m = 14$, it is noticed that decryption correctness condition (51) is valid using private key (f, g) as shown in (16) of Figure 4, and not valid for (F_i, G_i) returned by GLR as shown in (17) of Figure 4. Hence, GLR attack fails to return keys usable for ciphertext decryption.

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

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

$mgLen := 16 :$
$qLen := 80 :$
$q := 2^{qLen} :$
$f := 1351417702001 :$
$g := 2^{16} - 1 :$
$gcd(f, q \cdot g)$

$$1 \quad (1)$$

$is(g < 2^{mgLen} \textbf{ and } g \geq 2^{mgLen-1})$

$$true \quad (2)$$

$alpha := 1.07 :$
$is\left(f < 2^{qLen - mgLen - 1} \textbf{ and } f \geq alpha \cdot 2^{\frac{qLen}{2}}\right)$

$$true \quad (3)$$

$igcdex(f, q, 'Fq', 'qz') :$
$igcdex(f, g, 'Fg', 'gz') :$
Fq

$$154260404770580979079825 \quad (4)$$

Fg

$$2291 \quad (5)$$

$h := Fq \cdot g \textbf{ mod } q;$

$$417923022495305103287663 \quad (6)$$

$GLR(f, g, h, q)$

$$0$$

$$\begin{bmatrix} 1133958117 & 459459339518 & 26960316553 & 459459339518 & 891958362483 \\ 44728059201205 & 894561206306 & 2683683553383 & 894561206306 & 894561140771 \end{bmatrix} \quad (7)$$

$counter$

$$18 \quad (8)$$

$maxF := \max(result[1, 1..ColumnDimension(result)])$

$$891958362483 \quad (9)$$

$minG := \min(result[2, 1..ColumnDimension(result)])$

$$894561140771 \quad (10)$$

$rmin := ceil\left(\frac{q + g \cdot maxF}{minG}\right)$

$$1351417832690 \quad (11)$$

$rmax := floor\left(\frac{q}{g} - f\right)$

$$18447024201563593104 \quad (12)$$

$\max\left(\text{trunc}\left(\text{alpha} \cdot 2^{\frac{qLen}{2}}\right), rmin\right)$

$$1351417832690 \quad (13)$$

$is\left(rmax > 2 \cdot \max\left(\text{trunc}\left(\text{alpha} \cdot 2^{\frac{qLen}{2}}\right), rmin\right)\right)$

$$true \quad (14)$$

$r := rmin + 1024;$

$$1351417833714 \quad (15)$$

$m := 14 :$
$is(|r \cdot g + f \cdot m| < q)$

$$true \quad (16)$$

$F1 := result[1, 1] : F2 := result[1, 2] : F3 := result[1, 3] : F4 := result[1, 4] : F5 := result[1, 5] :$
$G1 := result[2, 1] : G2 := result[2, 2] : G3 := result[2, 3] : G4 := result[2, 4] : G5 := result[2, 5] :$

$is(|r \cdot G1 + F1 \cdot m| < q \textbf{ or } |r \cdot G2 + F2 \cdot m| < q \textbf{ or } |r \cdot G3 + F3 \cdot m| < q \textbf{ or } |r \cdot G4 + F4 \cdot m| < q \textbf{ or } |r \cdot G5 + F5 \cdot m| < q)$

$$false \quad (17)$$

$is(h \cdot rmin > q)$

$$true \quad (18)$$

Figure 4. Screenshot of the Code 2 run on Example 4.

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

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 6.3, Hybrid lattice basis reduction and MITM [52] in Section 6.2, Multiple transmission (MTA) [7] in Section 6.4, and also, the most recent, Chosen ciphertext [53–56], in Section 6.5) and tried applying them to RCPKC. Herein, NTRU parameters used, EES401EP1 [41], of the security level, $k = 112$ bits:

$$N = 401, p = 3, q = 2048, df_1 = df_2 = 8,$$

$$df_3 = 6, dg = 133, dr_1 = dr_2 = 8, dr_3 = 6. \tag{55}$$

In order to meet the same security level, the RCPKC settings satisfying (40) are:

$$qLen = 473, mgLen = 225. \tag{56}$$

The key space cardinality (defined in Section 6.1 for parameters (55), (56)) is greater or equal to $2^{2 \cdot k}$ for k =112 to avoid MITM attack explained in Section 6.1.

6.1. Brute Force and MITM Attacks

An attacker can recover 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 (9)). 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 (9)) has small coefficients. Equations (57) and (58) show search space cardinalities for g and f for security level, k =112 (taking into account MITM attack explained later in this section). Search space cardinality for f is computed as follows (see [53, Section 7]):

$$\begin{aligned} C_{NTRU}(f,k) &= \binom{N}{df_1}\binom{N-df_1}{df_1}\binom{N}{df_2}\binom{N-df_2}{df_2}\binom{N}{df_3}\binom{N-df_3}{df_3} \\ &= \binom{401}{8}\binom{393}{8}\binom{401}{8}\binom{393}{8}\binom{401}{6}\binom{395}{6} \\ &= 1.16 \times 10^{90} \geq 2^{2 \cdot k} = 2^{224}. \end{aligned} \tag{57}$$

Similarly, for g:

$$\begin{aligned} C_{NTRU}(g,k) &= \binom{N}{dg}\binom{N-dg}{dg} \\ &= \binom{401}{133}\binom{268}{133} = 4.34 \times 10^{188} \geq 2^{2 \cdot k} = 2^{224}. \end{aligned} \tag{58}$$

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 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 k =112 security level, NTRU parameters must be chosen to meet k = 224 security level as it is already made in (55). For RCPKC, secret value, g, is selected from the interval $[2^{mgLen-1}, 2^{mgLen})$ (see (35)), hence, search space cardinality for g to meet $2 \cdot k$-bit security level against brute force attack shall satisfy:

$$C_{RCPKC}(g,k) = 2^{mgLen-1} \geq 2^{2 \cdot k}. \tag{59}$$

The secret value, f, is selected from the interval $[\alpha \cdot 2^{qLen/2}, 2^{qLen-mgLen-1})$ (see (38)), hence, search space cardinality for f to meet $2 \cdot k$-bit security level against brute force attack shall satisfy:

$$C_{RCPKC}(f,k) = 2^{qLen-mgLen-1} - \alpha \cdot 2^{qLen/2} \geq 2^{2 \cdot k}. \quad (60)$$

For parameters (56), $C_{RCPKC}(g,k) = 2^{224}$, while $C_{RCPKC}(f,k) \approx 2^{247}$. In order to provide security level for $k = 112$, parameters (56) are chosen to meet twice greater security level of $2 \cdot k = 224$ to counter MITM attack, considered below, which reduces the brute force attack effort by 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, MITM attack can be applied to 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 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, RCPKC parameters (56) provide security level k=112 against brute force attack with MITM. Now let us consider 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, attacker can compromise 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 (35).

RCPKC message search space is defined by interval $[0, 2^{mgLen-1})$ (see(35)), hence, search space cardinality for m to meet $2 \cdot k$-bit security level against brute force attack shall satisfy:

$$C_{RCPKC}(m,k) = 2^{mgLen-1} \geq 2^{2 \cdot k}, \quad (61)$$

while the search space of r is defined by (48), (52), (53). Hence, search space cardinality for r to meet $2 \cdot k$-bit security level against brute force attack shall satisfy:

$$C_{RCPKC}(r,k) = rmax - max(\alpha \cdot 2^{qLen/2}, rmin) \geq 2^{2 \cdot k}. \quad (62)$$

Table 1 shows $mgLen$ and $qLen$ values to meet different $2 \cdot k$-bit security levels condition (62) (see Row 1, 2) and the width of the range for r (Row 7) with f and g specified in Rows 3, 4 respectively. It proves that the method can be practically used.

6.2. A Hybrid Lattice Basis Reduction and MITM Attack

The attack [52] on NTRU secret key combines 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 vectors from the other parts are just enumerated, thus combining concepts of LBRA and MITM attacks. The hybrid attack is not applicable to RCPKC since

- RCPKC lattice is 2-dimensional and cannot be split into the three subparts;
- RCPKC uses large-norm secret (f, g) vector (see (35), (38)) that cannot be found by LBRA looking for an SV, and the SV cannot be used for correct decryption (see (50)).

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632

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

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

6.3. Lattice Basis Reduction Attacks

The NTRU lattice basis, L_h^{NTRU}, associated with public key h defined in (2) 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 $h_0, ..., h_{N-1}$ are coefficients of the polynomial h. For convenience, matrix L_h^{NTRU} is abbreviated as

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

NTRU private key recovery problem can be formulated as 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

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

So, the vector (f,g) is in the lattice L_h^{NTRU}. Vector (f,g) or its rotation[1] can be found if it is the shortest vector in L_h^{NTRU}. 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[2]-years to find the shortest vector or its rotation for $N = 400$ (as in (55)) that most likely is the NTRU private key part, (f,g).

Contrary to NTRU, RCPKC is resistant to LBRA since 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.

6.4. Multiple Transmission Attack (MTA)

MTA reveals 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 (12) (see Section 3.1) is

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

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

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

thereby recovering $r^i - r^1 \bmod q$, $i = 1,...,n$, 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 (see (3)).

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 is presented where an attacker collects a large number of decryption failures. Another CCA is presented in [56] which is more efficient than [55] but still depends on decryption failures. RCPKC works on non-structured integers, and parameters, set in Section 5, guarantee correct decryption. Thus, neither of CCAs described above is applicable to RCPKC.

7. RCPKC One-Way Function and Semantic Security

7.1. RCPKC One-Way Function, Problem, and Assumption

Definition 1. *Negligible function [59, p. 56]. A function v: $\mathbb{N}\to\mathbb{R}$ is negligible if for every integer $c\geq 0$ there exists an integer k_c such that for all $k > k_c$,*

$$|v(k)| < \frac{1}{k^c}.$$

Definition 2. *Polynomial-time algorithm [60]. A polynomial-time algorithm, A, is one whose running time grows as a polynomial function of the input size.*

---

[1] Rotation of a polynomial, f, by i steps is $x^i\cdot f\in R_q$ for an integer i

[2] MIPS: Million Instructions Per Second

Definition 3. *Probabilistic algorithm [59, p. 54]. A probabilistic algorithm is algorithm having access to a source of random bits and, thus, it can make random choices.*

Definition 4. *Probabilistic polynomial-time [36, p. 3]. Probabilistic polynomial-time (PPT) adversary is an adversary that runs a PPT algorithm to attack a cipher.*

In this section, we prove the security of RCPKC one-way function based on the discussions of security of NTRU one-way function in [36]. According to Section 5.2 and Section 5.4, RCPKC defines the following four spaces:

- $\mathcal{D}_f = [\alpha \cdot 2^{qLen/2}, 2^{qLen-mgLen-1})$: 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.

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 (OW) function underlying RCPKC is:

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

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

Definition 5. *RCPKC-OW Problem. For a parameter set, $\mathcal{P}$, we denote by $Succ^{OW}_{RCPKC}(\mathcal{A}, \mathcal{P})$, the success probability of a PPT adversary, $\mathcal{A}$, for finding a pre-image of F_h,*

$$Succ^{OW}_{RCPKC}(\mathcal{A}, \mathcal{P}) = \Pr\begin{pmatrix}(m', r') \leftarrow \mathcal{A}(e, h) \\ s.t.\ F_h(m', r') = e\end{pmatrix}.$$

Assumption 1. *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*

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

7.2. Security of the RCPKC-OW Function

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, $Succ^{OW}_{RCPKC}(\mathcal{A}_1, \mathcal{P})$ is

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

Since, $qLen > mgLen$ by definition (40), $Succ^{OW}_{RCPKC}(\mathcal{A}_1, \mathcal{P})$ decreases exponentially in $qLen$, and Assumption 1 holds. Similarly, the attacker can try the following methods with exponentially decreasing success probability:

1. 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. 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. 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. Also, adversary can apply 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 above attacks.

7.3. RCPKC Semantic Security

In this section, we prove the indistinguishability against chosen plaintext attack (IND-CPA), i.e. semantic security of the RCPKC. The semantic, IND-CPA, security for RCPKC is considered based on the discussions of semantic IND-CPA security for NTRU [7] and results of [38].

Definition 6. *Public-Key CPA Indistinguishability Experiment [38, p. 24]. Let $\mathcal{E}$ be a public key encryption algorithm, $\mathcal{A}$ a PPT adversary. A public-key CPA indistinguishability experiment, $\boldsymbol{EXP}_{A,\mathcal{E}}^{IND-CPA}(k)$, is formalized as follows:*

1. *The user generates public and secret keys; public key is sent to $\mathcal{A}$.*
2. *$\mathcal{A}$ outputs two distinct valid messages m_0 and m_1, and sends the both to the user.*
3. *The user randomly chooses a bit, $b \in \{0,1\}$, and encrypts by $\mathcal{E}$ message m_b, getting a cipher, c. Then, the cipher, c, is sent to $\mathcal{A}$.*
4. *$\mathcal{A}$ outputs b' as a guess on b.*

The output of the experiment is 1 if $b' = b$, and 0, otherwise.

Definition 7. *IND-CPA security, Semantic security [38, p. 24]. A public-key encryption scheme, $\mathcal{E}$, is IND-CPA secure if, for every PPT adversary, $\mathcal{A}$, there exists a negligible function, $v_A(k)$, where k is a security parameter, such that*

$$\Pr(\boldsymbol{EXP}_{A,\mathcal{E}}^{IND-CPA}(k) = 1) \leq \frac{1}{2} + v_a(k).$$

It can be shown that breaking RCPKC IND-CPA security as it is done for NTRU [42, p. 3] is not possible because h is noninvertible due to (37), (54). Hence, we introduce Assumption 2.

Assumption 2. *RCPKC is IND-CPA secure.*

Let us show that Assumptions 1, 2 are equivalent. In the next discussions (definitions, theorems, proofs), we follow [38] with necessary changes.

Definition 8. *RCPKC Ciphertext Distribution Problem. Given the RCPKC public key, h, the RCPKC ciphertext distribution problem is to distinguish the following distributions: $PD_{UNI} = \{s \leftarrow_R \mathbb{Z}_q\}$ and $PD_{RCPKC} = \{s = h \cdot r \bmod q | r \leftarrow_R \mathcal{D}_r\}$, where $\leftarrow_R$ denotes random sampling from the respective domain.*

Theorem 1. *RCPKC ciphertext distribution problem is equivalent to RCPKC-OW problem.*

Proof. It consists of two parts considered below.
Part 1. If RCPKC ciphertext distribution problem can be solved, then adversary can solve RCPKC-OW problem.
Proof of Part 1. Let an adversary has a ciphertext, $c = h \cdot r + m \bmod q$, of the plaintext $m \in \mathcal{D}_m$. It is necessary disclosing (r, m). Set $i = 0, cl_0 = 0, cu_0 = 2^{mgLen}$.

Binary search step. Set $\delta_i = \frac{cl_i + cu_i}{2}$, $c^+ = c + \delta_i \bmod q$, and $c^- = c - \delta_i \bmod q$. Let the RCPKC ciphertext distribution solver returns 1, if its input is from PD_{RCPKC}, and 0, if its input is from PD_{UNI}. For (c^+, c^-) used as the input to the solver, it can return $Output = (a, s) \in (0,1)^2$, where a is the result of classification of c^+, and s is the result of classification of c^-. Note that by the RCPKC definition, c^+ is the ciphertext of the valid plaintext, $m + \delta_i$, if $m + \delta_i \in \mathcal{D}_m$, and not a valid ciphertext, otherwise. Similarly, c^- is the ciphertext of the valid plaintext, $m - \delta_i$, if $m - \delta_i \in \mathcal{D}_m$, and not a valid ciphertext, otherwise.

The $Output = (0,0)$ is not possible since $m \in \mathcal{D}_m$, and, hence, either the sum, or the difference shall be from $\mathcal{D}_m$, and, hence, respective ciphertext shall be recognized as belonging to PD_{RCPKC}. Thus, we have a contradiction, meaning that classification fails, and it shall be repeated. Because the solver is assumed having sufficient advantage, this variant is not expected happening often.

If $Output = (0,1)$, then $m \in [cl_{i+1}, cu_{i+1})$, which is $[2^{mgLen-1}, 2^{mgLen})$ for $i = 0$. Actually, if $Output = (0,1)$, subtraction of δ_i results in encryption of a number from $\mathcal{D}_m$, and, hence, the second ciphertext is considered as correct, but not the first one, for which the sum is out of $\mathcal{D}_m$.

If $Output = (1,0)$, then $m \in [cl_{i+1}, cu_{i+1}) = [cl_i, cu_i - \delta_i)$, which, in the case $i = 0$, is $[0, 2^{mgLen-1})$. Actually, if $Output = (1,0)$, addition of δ_i results in encryption of a number from $\mathcal{D}_m$, and, hence, the first ciphertext is considered as correct, but not the second one, for which the difference is out of $\mathcal{D}_m$.

The $Output = (1,1)$ is not possible since $m \in \mathcal{D}_m$, and, hence, either the sum, or the difference shall be not in $\mathcal{D}_m$, and, thus, respective ciphertext shall be recognized as not belonging to PD_{RCPKC}. Thus, we have a contradiction, meaning that classification fails, and it shall be repeated. Because the solver is assumed having sufficient advantage, this variant is not expected happening often.

In the result of the experiment described above, the uncertainty of the plaintext is twice decreased: we know, in which part of $\mathcal{D}_m$, junior or senior, m resides. Continuing the process by setting $i = i + 1$, and repeating the Binary search step, we come to the exact m value in at most $mgLen$ steps.

Part 2. If RCPKC-OW problem can be solved, then the RCPKC ciphertext distribution problem can be solved.

Proof of Part 2. Let s is given, and it is necessary deciding, whether it is from PD_{UNI}, or PD_{RCPKC}. Consider $c = s + m \bmod q$, where $m \in \mathcal{D}_m$, and ask the adversary to solve RCPKC-OW. If r is from $\mathcal{D}_r$, then s is from PD_{RCPKC}, otherwise, from PD_{UNI}. □

Theorem 2. *An adversary A can break the semantic security of RCPKC if and only if he can solve RCPKC distribution problem.*

Proof. It consists of two parts considered below.

Part 1. An adversary $\mathcal{A}$ can break the semantic security of RCPKC if he can solve RCPKC distribution problem.

Proof of Part 1. Adversary $\mathcal{A}$, generates two messages m_0 and m_1 from $\mathbf{D}_m$ such that $m_0 - m_1 \notin \mathcal{D}_m$, and sends both of them to challenger, $\mathcal{B}$. Challenger, $\mathcal{B}$, randomly chooses $b = 0$ or 1, encrypts $m_b \in \mathcal{D}_m$, by RCPKC to obtain ciphertext $c = r \cdot h + m_b \bmod q$, and sends c to adversary, $\mathcal{A}$. Having $c = r \cdot h + m_b \bmod q$, $\mathcal{A}$ finds $t = c - m_0 \bmod q$, which is $r \cdot h \bmod q \in PD_{RCPKC}$, if $b = 0$. On the other hand, $t = c - m_0 \bmod q = r \cdot h + m_1 - m_0 \in PD_{UNI}$, if $b = 1$. since $m_0 - m_1 \notin \mathcal{D}_m$. Hence, if the result is from PD_{RCPKC}, then c is the cipher of m_0, otherwise, of m_1, and the semantic security is broken.

Part 2. If an adversary, $\mathcal{A}$, can break the semantic security of RCPKC, then he can solve RCPKC distribution problem.

Proof of Part 2. Assume that the adversary, $\mathcal{A}$, can break the semantic security with an advantage greater than ϵ within a polynomial time. It means that, given two distinct m_0 and m_1 from $\mathcal{D}_m$, and a challenge ciphertext $c = r \cdot h + m_b \bmod q$, the adversary can succeed in guessing which message is encrypted with a probability at least $\frac{1}{2} + \epsilon$. Now, we use the adversary $\mathcal{A}$ as an oracle to derive an algorithm $\mathcal{B}$, which solves the RCPKC distribution problem with an advantage greater than ϵ and within a polynomial time. The input of the algorithm $\mathcal{B}$ is $s \in \mathbb{Z}_q$. The algorithm $\mathcal{B}$ outputs 0 or 1. By 0 (1, respectively), we denote the fact that the algorithm $\mathcal{B}$ decides that s comes from the distribution PD_{UNI} (PD_{RCPKC}, respectively). We construct the algorithm $\mathcal{B}$ as follows. Firstly, $\mathcal{B}$ runs the algorithm $\mathcal{A}$ to output two plaintexts m_0 and m_1 from $\mathcal{D}_m$ chosen by the adversary $\mathcal{A}$. Then the algorithm $\mathcal{B}$ randomly chooses $b = 0$ or 1, and computes $c = s + m_b \bmod q$. Then the algorithm $\mathcal{B}$ feeds the challenge ciphertext c into the adversary, $\mathcal{A}$, and $\mathcal{A}$ outputs a value d as the guess of the algorithm $\mathcal{A}$ on the algorithm $\mathcal{B}$'s choice of b. If $d = b$, the algorithm $\mathcal{B}$ outputs 1; otherwise, the algorithm $\mathcal{B}$ outputs 0. Now we compute the probability for the algorithm $\mathcal{B}$ to succeed in distinguishing both distributions PD_{UNI} and PD_{RCPKC}. If s comes from the random distribution PD_{UNI}, we compute the probability $\Pr(0)$ for the algorithm $\mathcal{B}$ to succeed in deciding that s comes from the distribution PD_{UNI}, namely, the probability that the algorithm $\mathcal{B}$ outputs 0. According to our construction of the algorithm

$\mathcal{B}$, the probability is exactly the probability that $d \neq b$. In other words, the probability $\Pr(0)$ should be the failure probability for the algorithm $\mathcal{A}$ to guess b. Noting that s is uniformly distributed over PD_{UNI}, we conclude that $c = s + m_b \bmod q$ is also uniformly distributed over PD_{UNI}, and hence is independent of the choice of b. Hence, the probability for the adversary $\mathcal{A}$ to succeed in guessing the value of b is exactly $\frac{1}{2}$. So, the failure probability for the algorithm $\mathcal{A}$ in guessing b is also exactly $\frac{1}{2}$. So, we conclude that $\Pr(0) = \frac{1}{2}$.

Consider now the case that s comes from PD_{RCPKC}. The probability, $\Pr(1)$, for the algorithm $\mathcal{B}$ to succeed in deciding that s comes from PD_{RCPKC}, is the probability for the algorithm $\mathcal{B}$ to output 1. That is, $\Pr(1)$, should be the probability for the algorithm $\mathcal{A}$ to succeed in obtaining the right value of b (the probability for $d = b$). Recalling that s comes from PD_{RCPKC}, we conclude that there must exist $r \in \mathcal{D}_r$, such that $s = r \cdot h \bmod q$ is from PD_{RCPKC}. So $c = s + m_b \bmod q$ is a valid ciphertext of m_b. So the adversary $\mathcal{A}$ guesses the exact value b with a success probability greater than $\frac{1}{2} + \epsilon$, thus, $\Pr(1) \geq \frac{1}{2} + \epsilon$. So the advantage of the algorithm $\mathcal{B}$ in successfully distinguishing the distributions PD_{UNI} and PD_{RCPKC} is $|\Pr(1) - \Pr(0)| \geq \epsilon$. The algorithm $\mathcal{B}$ only performs once the oracle $\mathcal{A}$ and makes addition $s + m_b \bmod q$, taking polynomial time. □

From Theorems 1, 2 immediately follows

Theorem 3. *RCPKC is IND-CPA secure if and only if Assumption 2 holds.*

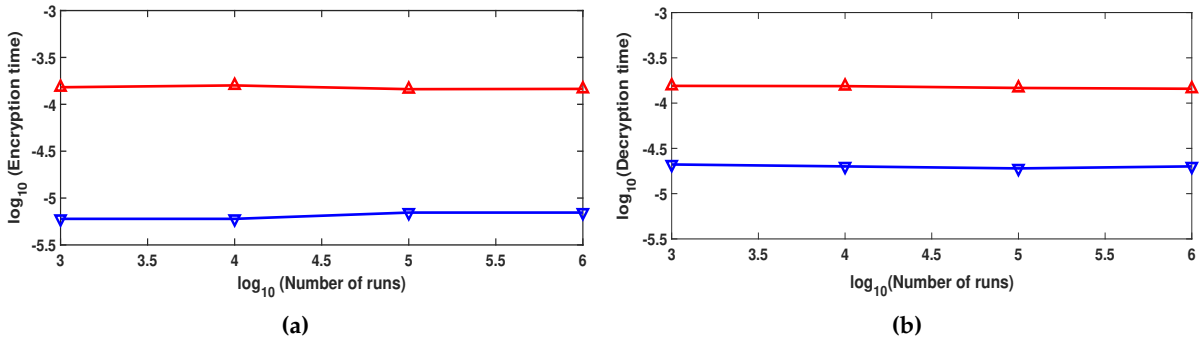
8. RCPKC Performance and Power Consumption Evaluation

8.1. RCPKC Performance Evaluation

Experiments were conducted using NTRU code [61], and RCPKC implementation in C99 language similar to [61] with NTL library [62] on a PC equipped with 2 GHz Intel Pentium Dual CPU E2180, 3 GB RAM, and Windows 10. Both NTRU code [61] and the proposed RCPKC are implemented in Visual Studio 2017. NTRU parameters (55) and RCPKC parameters (56) are used. CPU encryption and decryption time of RCPKC and NTRU is measured for 10^3, 10^4, 10^5, and 10^6 runs (see Figures 5) with respective averages. In each run, new secret and public keys, and messages are chosen randomly for NTRU and RCPKC. From Figures 5a, 5b, it is observed that RCPKC is 23.34 times faster than NTRU in encryption, and 7.5 times faster in decryption respectively. The large difference in the RCPKC encryption/decryption time is due to the observation made in our experiments that multiplication time depends on the length of the operands, and the greater-length operands are used in the decryption. In comparison to NTRU variants presented in Section 1, BQTRU [19] is faster than NTRU in encryption/decryption by seven times, ETRU [25] is faster than NTRU in encryption/decryption by 1.45/1.72 for $N = 400$, and MaTRU is faster than NTRU 2.5 times in encryption/decryption, while other NTRU variants introduced in Section 1 have not published information regarding their performance. It is observed that RCPKC is faster than the fastest most recent published NTRU variant, BQTRU, more than three times in encryption. Figure 6 compares RCPKC and NTRU variants encryption and decryption time.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 July 2020 doi:10.20944/preprints202007.0066.v1

Peer-reviewed version available at Sensors 2020, 20, 4632; doi:10.3390/s20164632



(a) (b)

Figure 5. CPU encryption (**a**) and decryption (**b**) time of RCPKC (blue) and NTRU (red) for 10^3, 10^4, 10^5, and 10^6 runs with respective averages.

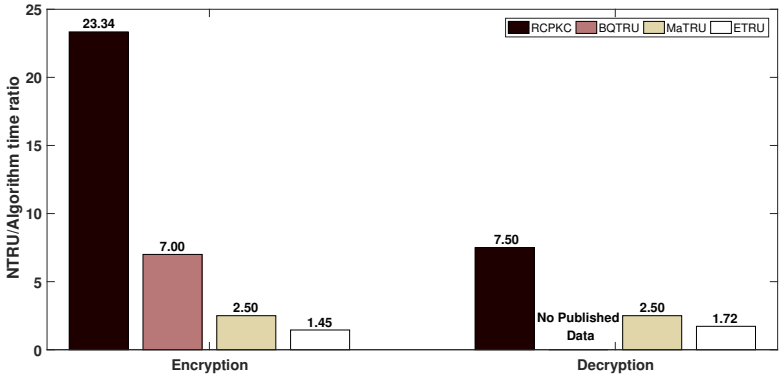


Figure 6. NTRU versus RCPKC, BQTRU, MaTRU, and ETRU encryption and decryption time ratio.

8.2. RCPKC Power Consumption Evaluation

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

Same frequencies. Let RCPKC and NTRU execution time be T_{RCPKC}, and T_{NTRU}, respectively. Then, from (20), the consumed energy by NTRU and RCPKC E_{NTRU} and E_{RCPKC} is:

$$\begin{aligned} E_{NTRU} &= P \cdot T_{NTRU}, \\ E_{RCPKC} &= P \cdot T_{RCPKC}. \end{aligned} \tag{63}$$

And, since T_{NTRU} is greater than T_{RCPKC} by more than 7 times, then, from (63):

$$\frac{E_{NTRU}}{E_{RCPKC}} = \frac{T_{NTRU}}{T_{RCPKC}} \geq 7. \tag{64}$$

From (64), RCPKC consumes seven times energy less than NTRU using the same frequency.

Different frequencies. Since RCPKC is 7.5 times faster than NTRU, the former takes approximately the same run time on eight times lower clock frequency CPU than that of the latter. Dynamic and leakage power consumption, calculated for frequencies from [48, p. 19] according to (23), are shown in Table 2:

Table 2. MSP430FR5969 microcontroller dynamic and leakage power consumption, P_{dyn}, and P_{leak} for frequencies from [48, p. 19] at active supply voltages 3 and 2.2 V.

	2.2 V		3 V	
frequency (MHz)	P_{dyn} **(**μ**W)**	P_{leak} **(nW)**	P_{dyn} **(**μ**W)**	P_{leak} **(nW)**
1	**48.4**	**44**	**90**	**60**
8	**387.2**		**720**	

It follows from Table 2 that $P_{leak} << P_{dyn}$, and can be neglected. From Table 2, it follows that reducing clock frequency from 8 to 1 MHz ($\frac{8}{1} = 8$ times) leads to eight times power consumption reduction from 720 to 90 μW. Note that MSP430FR5969, in lower frequency, operates at lower voltage: operating on 1 MHz frequency at 2.2 V [48, p. 19] results in 48.4 μ W dynamic power consumption. Hence, the total power reduction is $\frac{720}{48.4} \approx 15$ times.

Therefore, RCPKC, compared to NTRU, is better applicable to WSN with power constrained devices.

9. Conclusion

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 the LBRA because its private key components, f, g, are chosen big with respect to $\sqrt{q}$ to form a two-component vector with the norm exceeding the Minkowski's boundary (29)-(32) for the shortest vector in a two-dimensional lattice and meeting (33). Hence, LBRA by GLR algorithm returning the shortest vector in a two-dimensional lattice fails finding the large-norm private key vector, (f, g).

In spite of the big numbers, f, r, meeting (38) used in RCPKC, it guarantees decryption correctness condition (14) holding (see (41)) due to the use of conditions (35), (38), (40), (48), (52) instead of the conditions (6), (10), (11), used in the original insecure CPKC (see Sections 3.2-3.2.2) considered in [24]. It was found that insecurity of the original CPKC stems from the use of the conditions (6), (10), (11), defining smaller than $\sqrt{q}$ numbers f, g, m, r meeting Minkowski's boundary (29) and decryption correctness condition (14). RCPKC is resistant to LBRA by GLR attack due to the the special choice of the range for the random value, r, used in encryption (12) that guarantees correctness condition (14) violation for the short vectors returned by GLR but its holding for the original private key, (f, g). Section 6 shows also that security of RCPKC with respect to other known attacks on NTRU is not less than that of NTRU that allows us concluding that RCPKC is more secure than NTRU. Section 7 proves RCPKC IND-CPA security under the assumption of hardness of its OW function.

RCPKC uses numbers, i.e. minimal possible, degree zero, polynomials, that makes it about 24 (7) times more effective in encryption (decryption) than NTRU, and more than three times more effective in encryption with respect to the fastest most recent published NTRU variant, BQTRU [19], as experiments show (see Figures 5, 6). Compared to NTRU, RCPKC reduces energy consumption at least seven times that allows increasing life-time of unattended WSN more than seven times.

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

Author Contributions: Conceptualization, A.I. and A.C.; Funding acquisition, J.R.; Investigation, A I. and A.C.; Methodology, A.I., A. C. and N.H.; Software, J.R.; Validation, A.K.; Visualization, Y.D.; Writing – review & editing, Y.D., A.K. and Joel J.R.

Acknowledgments: A.I., N.H., and Y.D. would like to thank Palestine Technical University–Kadoorie for provided support. J.R.'s work is supported by FCT/MCTES through national funds and when applicable co-funded by EU

funds under the Project UIDB/EEA/50008/2020; and by Brazilian National Council for Research and Development (CNPq) via Grant No. 309335/2017-5. We thank anonymous reviewers for useful comments.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analysis, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Yick, J.; Mukherjee, B.; Ghosal, D. Wireless Sensor Network Survey. *Comput. Netw.* **2008**, *52*, 2292–2330. doi:10.1016/j.comnet.2008.04.002.
2. Akyildiz, I.F.; Melodia, T.; Chowdhury, K.R. A survey on wireless multimedia sensor networks. *Comput. Networks* **2007**, *51*, 921–960. doi:10.1016/j.comnet.2006.10.002.
3. Gungor, V.C.; Lu, B.; Hancke, G.P. Opportunities and Challenges of Wireless Sensor Networks in Smart Grid. *IEEE Trans. Industrial Electronics* **2010**, *57*, 3557–3564. doi:10.1109/TIE.2009.2039455.
4. Li, M.; Lou, W.; Ren, K. Data security and privacy in wireless body area networks. *IEEE Wirel. Commun.* **2010**, *17*, 51–58. doi:10.1109/MWC.2010.5416350.
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. doi:10.1016/j.ins.2015.02.010.
6. Feng, D.; Jiang, C.; Lim, G.; Jr., L.J.C.; Feng, G.; Li, G.Y. A Survey of Energy-Efficient Wireless Communications. *IEEE Communications Surveys and Tutorials* **2013**, *15*, 167–178. doi:10.1109/SURV.2012.020212.00049.
7. Hoffstein, J.; Pipher, J.; Silverman, J.H. NTRU: A Ring-Based Public Key Cryptosystem. Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings; Buhler, J., Ed. Springer, 1998, Vol. 1423, *Lecture Notes in Computer Science*, pp. 267–288. doi:10.1007/BFb0054868.
8. Hermans, J.; Vercauteren, F.; Preneel, B. Speed Records for NTRU. Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings; Pieprzyk, J., Ed. Springer, 2010, Vol. 5985, *Lecture Notes in Computer Science*, pp. 73–88. doi:10.1007/978-3-642-11925-5_6.
9. Kaur, I.; Kour, H.; Verma, A. Ticket based Secure Authentication Scheme using NTRU Cryptosystem in Wireless Sensor Network. *International Journal of Computer Science and Information Security* **2017**, *15*, 55.
10. 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. doi:10.1145/2700078.
11. 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. From Science to Society; Otjacques, B.; Hitzelberger, P.; Naumann, S.; Wohlgemuth, V., Eds.; Springer: Cham, 2018; pp. 181–189.
12. Malla, A.; Sahu, P.; Mishra, M. A Novel Encryption Scheme for Secure SMS Communication. Advances in Electronics, Communication and Computing; Kalam, A.; Das, S.; Sharma, K., Eds.; Springer: Singapore, 2018; pp. 467–478. doi:https://doi.org/10.1007/978-981-10-4765-7_50.
13. Jun, J.; Chen, H. A novel mutual authentication and key agreement protocol based on NTRU cryptography for wireless communications. *Journal of Zhejiang University-SCIENCE A* **2005**, *6*, 399–404. doi:10.1007/BF02839407.
14. Bailey, D.V.; Coffin, D.; Elbirt, A.J.; Silverman, J.H.; Woodbury, A.D. NTRU in Constrained Devices. Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings; Koç, Ç.K.; Naccache, D.; Paar, C., Eds. Springer, 2001, Vol. 2162, *Lecture Notes in Computer Science*, pp. 262–272. doi:10.1007/3-540-44709-1_22.
15. 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. Design, Automation & Test in Europe Conference & Exhibition, DATE 2017, Lausanne, Switzerland, March 27-31, 2017; Atienza, D.; Natale, G.D., Eds. IEEE, 2017, pp. 698–703. doi:10.23919/DATE.2017.7927079.

16. Kamal, A.A.; Youssef, A.M. Strengthening hardware implementations of NTRUEncrypt against fault analysis attacks. *Journal of Cryptographic Engineering* **2013**, *3*, 227–240. doi:10.1007/s13389-013-0061-7.
17. Sedenion. https://en.wikipedia.org/wiki/Sedenion, Last Accessed 29/01/2020.
18. Alsaidi, N.M.; Yassein, H.R. BITRU: Binary Version of the NTRU Public Key Cryptosystem via Binary Algebra. *International Journal of Advanced Computer Science and Applications* **2016**, *7*, 1–6.
19. Bagheri, K.; Sadeghi, M.R.; Panario, D. A non-commutative cryptosystem based on quaternion algebras. *Designs, Codes and Cryptography* **2017**. published online https://link.springer.com/journal/10623/onlineFirst/page/1, doi:10.1007/s10623-017-0451-4.
20. Banks, W.D.; Shparlinski, I.E. A Variant of NTRU with Non-invertible Polynomials. Progress in Cryptology - INDOCRYPT 2002, Third International Conference on Cryptology in India, Hyderabad, India, December 16-18, 2002; Menezes, A.; Sarkar, P., Eds. Springer, 2002, Vol. 2551, *Lecture Notes in Computer Science*, pp. 62–70. doi:10.1007/3-540-36231-2_6.
21. Coglianese, M.; Goi, B. MaTRU: A New NTRU-Based Cryptosystem. Progress in Cryptology - INDOCRYPT 2005, 6th International Conference on Cryptology in India, Bangalore, India, December 10-12, 2005, Proceedings; Maitra, S.; Madhavan, C.E.V.; Venkatesan, R., Eds. Springer, 2005, Vol. 3797, *Lecture Notes in Computer Science*, pp. 232–243. doi:10.1007/11596219_19.
22. Gaborit, P.; Ohler, J.; Solé, P. CTRU, a polynomial analogue of NTRU. PhD thesis, INRIA, 2002.
23. Gaithuru, J.N.; Salleh, M. ITRU: NTRU-Based Cryptosystem Using Ring of Integers. *International Journal of Innovative Computing* **2017**, *7*, 33–38.
24. Hoffstein, J.; Pipher, J.; Silverman, J.H. *An Introduction to Mathematical Cryptography*; Springer: New York, 2014; pp. 373–376. doi:10.1007/978-1-4939-1711-2_7.
25. Jarvis, K.; Nevins, M. ETRU: NTRU over the Eisenstein integers. *Designs, Codes and Cryptography* **2015**, *74*, 219–242. doi:10.1007/s10623-013-9850-3.
26. Karbasi, A.H.; Atani, R.E. ILTRU: An NTRU-Like Public Key Cryptosystem Over Ideal Lattices. *IACR Cryptology ePrint Archive* **2015**, *2015*, 549.
27. Malekian, E.; Zakerolhosseini, A. OTRU: A non-associative and high speed public key cryptosystem. Computer Architecture and Digital Systems (CADS), 2010 15th CSI International Symposium on. IEEE, 2010, pp. 83–90.
28. Malekian, E.; Zakerolhosseini, A.; Mashatan, A. QTRU: quaternionic version of the NTRU public-key cryptosystems. *The ISC International Journal of Information Security* **2011**, *3*, 29–42.
29. Thakur, K.; Tripathi, B. STRU: A Non Alternative and Multidimensional Public Key Cryptosystem. *Global Journal of Pure and Applied Mathematics* **2017**, *13*, 1447–1464.
30. Thakur, K.; Tripathi, B.P. BTRU, A Rational Polynomial Analogue of NTRU Cryptosystem. *International Journal of Computer Applications* **2016**, *145*, 22–24. doi:10.5120/ijca2016910769.
31. Vats, N. NNRU, a noncommutative analogue of NTRU. *arXiv preprint arXiv:0902.1891* **2009**.
32. Yassein, H.R.; Al-Saidi, N.M. HXDTRU Cryptosystem Based On Hexadecnion Algebra. Proceeding of the 5th International Cryptology and Information Security Conference, Kota Kinabalu, Malaysia, May 31 - Jun 02, 2016, 2016.
33. Yu, Y.; Xu, G.; Wang, X. Provably Secure NTRU Instances over Prime Cyclotomic Rings. Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part I; Fehr, S., Ed. Springer, 2017, Vol. 10174, *Lecture Notes in Computer Science*, pp. 409–434. doi:10.1007/978-3-662-54365-8_17.
34. 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.
35. Seck, M.; Sow, D. BI-NTRU Encryption Schemes: Two New Secure Variants of NTRU. Algebra, Codes and Cryptology; Gueye, C.T.; Persichetti, E.; Cayrel, P.L.; Buchmann, J., Eds.; Springer International Publishing: Cham, 2019; pp. 216–235.
36. Howgrave-Graham, N.; Silverman, J.H.; Whyte, W. Choosing Parameter Sets for NTRUEncrypt with NAEP and SVES-3. *IACR Cryptol. ePrint Arch.* **2005**, *2005*, 45.
37. 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.
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. doi:10.1016/j.ins.2018.01.037.

39. Lenstra, A.K.; Lenstra, H.W.; Lovász, L. Factoring polynomials with rational coefficients. *Mathematische Annalen* **1982**, *261*, 515–534.
40. Hoffstein, J.; Silverman, J.H.; Whyte, W. Estimated Breaking Times for NTRU Lattices. Technical report, version 2, NTRU Cryptosystems http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.10.6336&rep=rep1&type=pdf, 2003.
41. IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. *IEEE Std 1363.1-2008* **2009**, pp. 1–81. doi:10.1109/IEEESTD.2009.4800404.
42. Hoffstein, J.; Silverman, J.H. Protecting NTRU against chosen ciphertext and reaction attacks. Technical report, Technical report, NTRU Cryptosystems, 2000.
43. Ibrahim, A.; Chefranov, A.; Hamad, N. NTRU-Like Secure and Effective Congruential Public-Key Cryptosystem Using Big Numbers. 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS) , Amman, Jordan, October 9-11, 2019, pp. 20–26.
44. Smith, C. *Environmental physics*; Routledge: London, 2002; pp. 1028–1029. doi:10.1002/esp.396.
45. Beloglazov, A.; Buyya, R.; Lee, Y.C.; Zomaya, A.Y. A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. *Advances in Computers* **2011**, *82*, 47–111. doi:10.1016/B978-0-12-385512-1.00003-7.
46. Nikolic, B. Design in the Power-Limited Scaling Regime. *IEEE Transactions on Electron Devices* **2008**, *55*, 71–83.
47. Romli, N.; Minhad, K.; Reaz, M.; Amin, M.S. An overview of power dissipation and control techniques in cmos technology. *Journal of Engineering Science and Technology* **2015**, *10*, 364–382.
48. Texas Instruments Inc. *MSP430FR596x, MSP430FR594x Mixed-Signal Microcontrollers*, 2018. SLAS704G.
49. Bourbaki, N. *Topological Vector Spaces: Chapters 1—5*, 1st ed.; Elements of Mathematics, Springer-Verlag Berlin Heidelberg, 2003.
50. Maple. https://www.maplesoft.com/, Last Accessed 29/01/2020.
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: Verlag, Berlin, Heidelberg, 2010.
52. Howgrave-Graham, N. A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack Against NTRU. Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings; Menezes, A., Ed. Springer, 2007, Vol. 4622, *Lecture Notes in Computer Science*, pp. 150–169. doi:10.1007/978-3-540-74143-5_9.
53. Hoffstein, J.; Pipher, J.; Schanck, J.M.; Silverman, J.H.; Whyte, W.; Zhang, Z. Choosing Parameters for NTRUEncrypt. Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings; Handschuh, H., Ed. Springer, 2017, Vol. 10159, *Lecture Notes in Computer Science*, pp. 3–18. doi:10.1007/978-3-319-52153-4_1.
54. Jaulmes, É.; Joux, A. A Chosen-Ciphertext Attack against NTRU. Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings; Bellare, M., Ed. Springer, 2000, Vol. 1880, *Lecture Notes in Computer Science*, pp. 20–35. doi:10.1007/3-540-44598-6_2.
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. Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings; Boneh, D., Ed. Springer, 2003, Vol. 2729, *Lecture Notes in Computer Science*, pp. 226–246. doi:10.1007/978-3-540-45146-4_14.
56. Gama, N.; Nguyen, P.Q. New Chosen-Ciphertext Attacks on NTRU. Public Key Cryptography - PKC 2007, 10th International Conference on Practice and Theory in Public-Key Cryptography, Beijing, China, April 16-20, 2007, Proceedings; Okamoto, T.; Wang, X., Eds. Springer, 2007, Vol. 4450, *Lecture Notes in Computer Science*, pp. 89–106. doi:10.1007/978-3-540-71677-8_7.
57. Howgrave-Graham, N.; Silverman, J.H.; Whyte, W. A Meet-in-the-Middle Attack on an NTRU Private key. Technical report, Technical report, NTRU Cryptosystems, June 2003. Report, 2003.
58. Kirchner, P.; Fouque, P. Revisiting Lattice Attacks on Overstretched NTRU Parameters. Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I; Coron, J.; Nielsen, J.B., Eds., 2017, Vol. 10210, *Lecture Notes in Computer Science*, pp. 3–26. doi:10.1007/978-3-319-56620-7_1.

59. Katz, J.; Lindell, Y. *Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series)*; Chapman & Hall/CRC, 2007.
60. Kaliski, B., Polynomial Time. In *Encyclopedia of Cryptography and Security*; van Tilborg, H.C.A.; Jajodia, S., Eds.; Springer US: Boston, MA, 2011; pp. 948–949. doi:10.1007/978-1-4419-5906-5_425.
61. Whyte, W.; Etzel, M. Open Source NTRU Public Key Cryptography Algorithm and Reference Code. Last Accessed 29/01/2020.
62. Victor, S. NTL: A library for doing number theory. http://www. shoup. net/ntl/, Last Accessed 29/01/2020.