EASR

Engineering and Applied Science Research

https://www.tci-thaijo.org/index.php/easr/index

Published by the Faculty of Engineering, Khon Kaen University, Thailand

An asymmetric cryptography using Gaussian integers

Wanarat Juraphanthong and Suradet Jitprapaikulsarn*

Department of Electrical and Computer Engineering, Faculty of Engineering, Naresuan University, Phitsanulok 65000, Thailand

Received 7 May 2019
Revised 31 July 2019
Accepted 18 November 2019

Abstract

In this paper, the already strong McEliece cryptosystem is enhanced with a two-dimensional finite Gaussian integer. By substituting the one-dimensional linear code with a two-dimensional code employing a finite Gaussian integer, a new system simultaneously increases the key space and the errors to be correct by syndrome decoding. We compare the proposed system against the classic McEliece system in three aspects: the work factors performing the trial of the attacks, the computational complexity cost, and the empirical running time of the system. Compared to the classic McEliece cryptosystem, the enhanced cryptosystem achieves a higher security level against key recovering and decoding attacks. By carefully selecting parameters, a small code element can improve the key strength without compromising the runtime efficiency.

Keywords: Asymmetric cryptography, Code-based cryptosystem, McEliece cryptosystem, Gaussian integer

1. Introduction

It has been revealing that quantum algorithms implemented on quantum computers are capable of breaking the widely used asymmetric cryptosystems. The commonly-used cryptosystems that rely on the practical difficulty of the factorization of the product of two large prime numbers or the difficulty of inverting a discrete logarithm problem can be broken by Shor's algorithm [1]. Quantum computers appear to be only concepts at the moment. However, it is expected that functioning devices will be built in the upcoming decades. To prepare for this inevitable future, cryptography that can resist a quantum-algorithm-based attack called post-quantum cryptography (PQC) is being proposed. The classic McEliece cryptography [2] is a code-based PQC using the binary Goppa codes as an error correcting code. It relies on the difficulty of the syndrome decoding problem of a linear code. In detail, the syndrome decoding problem is an NP-complete problem that cannot be conveniently solved by quantum algorithms. The key advantage of the system is that both algorithms for encryption and decryption have low complexity. However, the large key size is the main weakness and the problem to be resolved by researchers.

Several improvements focused on substituting the Goppa codes with other one-dimensional linear codes. Niederreiter [3] proposed a digital signature scheme based on McEliece using Reed-Solomon code. The scheme can reduce the key size, but it is broken by structural attacks [4]. Sidelnikov [5] proposed a Reed-Muller coding-based cryptosystem with a lower key length to obtain faster encryption and decryption. However, the system has been cryptanalyzed with an observation attack [6]. Monico et al. [7] replaced Goppa codes with low-density parity check (LDPC) code to decrease the key size, but the permutation matrix was unable to hide the secret key because of the low-density weight of the scheme. In another improvement on resolving the permutation problem, Baldi et al. [8] considered a quasi-cyclic low-density parity check (QC-LDPC) code and Misoczki et al. [9] introduced a moderate density parity check (MDPC) code. A QC-LDPC cryptosystem is needed to extend the specific matrices, but the MDPC cryptosystem did not require any extended matrices. However, both cryptosystems failed during the decryption process. Shrestha and Kim [10] introduced a code-based cryptosystem using a polar code that can reduce the decryption key length. Furthermore, Hooshmand et al. [11] suggested a public key scheme based on polar codes to decrease the key pair lengths. Polar code-based schemes were successfully cryptanalyzed by Bardet et al. [12]. Hooshmand et al. [13] introduced the PKC-PC, the new public key cryptosystem using polar code that employs a random selection of subcode to prevent this type of cryptanalyst attack.

The use of Gaussian integers was one of the more efficient approaches to increase the security level in widely used asymmetric cryptosystems. For example, Pradhan and Sharma [14] proposed a modified algorithm using Gaussian integers in an RSA cryptosystem. This scheme provided more security compared to the classical approach because it increased the number of chosen messages and public exponents that made trials more complicated. Mohamed and Elkamchouchi [15] proposed a new approach to improve

*Corresponding author. Tel.: +66 5596 4321
Email address: suradetj@gmail.com
doi: 10.14456/easr.2020.16

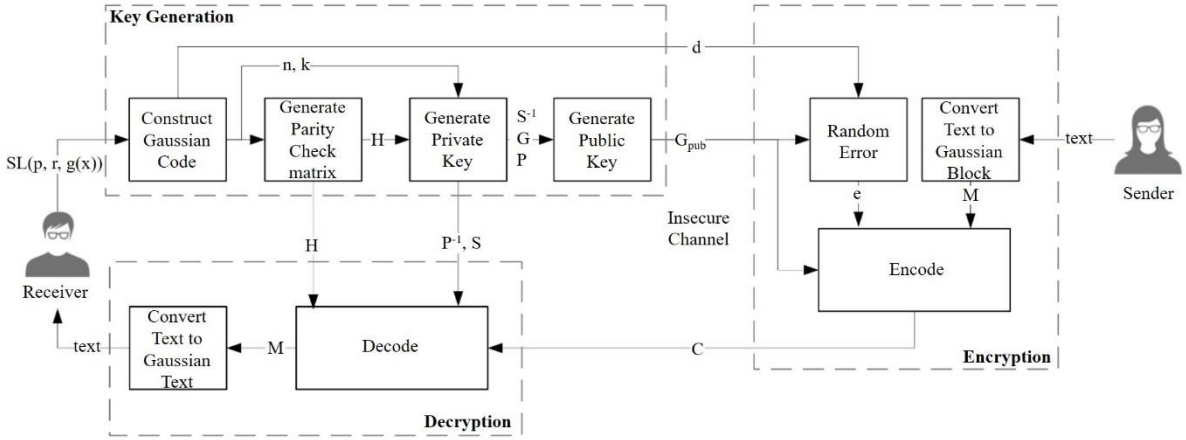

Figure 1 The framework of the proposed cryptosystem

security in elliptic curve cryptography. The reason to use Gaussian integers instead of rational integers is to acquire a greater number of points to increase the strength of the curve. In PQC, Nanda et al. [16] modified the NTRU cryptography algorithm using a Gaussian integer matrix. This method increased the security due to expansion of the key space.

Prior to our study, there has been no publication on applying multi-dimensional error correcting code with the McEliece cryptosystem. In this paper, we propose a novel McEliece-based cryptosystem using the two-dimensional Mannheim code over Gaussian integer fields instead of the one-dimensional linear code used in the classic McEliece cryptosystem. The Mannheim code increases the range of code words and syndrome of code. Hence, it improves the security level of the cryptosystem.

2. Methodology

2.1 Codes over Gaussian integers

Mannheim error correcting (MEC) codes are two-dimensional coding systems over Gaussian integers designed by Huber [17-18]. MEC codes are constructed based on finite Gaussian integer fields (G_π) and are corrected in accordance with the Mannheim distance. In detail, the G_π are constructed from prime p of the form $p \equiv 1 \bmod 4$ that $p = \pi \cdot \pi^*$ where π and π^* are a complex number and its conjugate, respectively. The Mannheim distance shows the difference between two elements of G_π which can be composed by Mannheim weight.

2.1.1 Constructing the generator and parity-check matrices

We start by designing a MEC of length $n=(p^r-1)/4$ which contains elements α in G_π of order p. Then, the parity-check matrix H and the corresponding generator matrix G can be constructed as Eq. (1) and (2), respectively:

$$H = [1, \alpha^r, \alpha^{r+1}, \ldots, \alpha^{((p^r-1)/4)-1}] \quad (1)$$

$$G = \begin{bmatrix} -\alpha^r & 1 & 0 & \ldots & 0 \\ -\alpha^{r+1} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\alpha^{((p^r-1)/4)-1} & 0 & 0 & \ldots & 1 \end{bmatrix} \quad (2)$$

For example, from [17], with a sample prime $p = 13$, $\pi = 3+2i$, $r=1$ and $\alpha = 1+i$, we can construct the matrices $H = [1, 1+i, 2i]$ and $G = \begin{bmatrix} -(1+i) & 1 & 0 \\ -2i & 0 & 1 \end{bmatrix}$.

2.1.2 MEC encoding and decoding

In MEC encoding, the generator matrix G encodes input vectors to code word vectors $c = (c_0, c_1, \ldots, c_{n-1})$ over G_π. The code words are sent across a channel with noise, hence different syndromes of code words are produced by any error vectors. The important purpose of the decoding process is to correct the received vector $r = c+e$. The parity-check matrix H is used to compute the value of errors and to discover their location occurring in the syndromes of the received vector. The syndromes can be computed by $s = H \cdot r^T$. Their positions are given by $l = \log_\alpha s \bmod n$ and the values by $s \cdot \alpha^{-l}$. In the previous example, we assume that a vector $r = (1+i, i, -1+i)$. Then the syndrome $s = -2 = \alpha^{11}$, the location $l = 11 \bmod 3 = 2$, the error value is $-2 \cdot \alpha^{-2} = i$, and we have the error vector $e = (0, 0, i)$. Therefore, the code word is $c = r-e$ is $(1+i, i, -1)$.

2.2 Proposed cryptosystem

The proposed cryptosystem is based on McEliece using MEC over Gaussian integers. Briefly, the sender wants to transmit messages to a receiver. The receiver assigns the security level (SL) that variables p, r, and primitive polynomial g(x) will be set automatically in the cryptosystem for the initial state. The system generates a private key and a public key for encoding and decoding processes. The text messages are encoded to ciphertexts and sent across an unsecured channel. Then ciphertexts are decoded to plaintexts using the private key. The framework of the cryptography system is shown in Figure 1.

2.2.1 Key generation

In key generation, the security parameters n, k, d, and p are automatically generated from security level parameters. The cryptosystem constructs $[n, k, d_M]p$ MEC of primitive element L that are the block codes over finite Gaussian integer fields G_{π^r} with the length of $n=(p^r-1)/4$ symbols, dimension of $k = n-r$ symbols and minimum Mannheim distance d_M. Then, the codes define the parity check matrix H with corresponding generator matrix G for correcting syndrome codes in the decryption process and for generating the private key, respectively.

The private key includes the secret generator matrix G and two non-singular matrices, a scramble matrix S and permutation matrix P. We use elements in G_π to random

Table 1 Key generation algorithm

Algorithm 1 Key generation
Input: The security level SL = (p, r, g(x)) where $p \equiv 1 \bmod 4$, $g(x) \in G_\pi[x]$ with $\deg(g(x)) = r$
Output: The public key G_{pub} and public variables n, k, d, and p; The private key H, P, and S
1: Construct $[n, k, d_M]p$ MEC codes L that $n = (p^r-1)/4$,k = n-r and d_M = minimum Mannheim distance
2: Generate $H = [1, \alpha^r, \alpha^{r+1}, \ldots, \alpha^{n-1}]$, $\alpha \in L$
3: Generate $G = \begin{bmatrix} -\alpha^r & 1 & 0 & \ldots & 0 \\ -\alpha^{r+1} & 0 & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\alpha^{n-1} & 0 & 0 & \ldots & 1 \end{bmatrix}$, $GH^T = 0$
4: Generate a scramble $S \in L$ of dimension k x k
5: Generate a permutation P of dimension n x n
6: $G_{pub} = S^{-1}GP \bmod \pi$
7: **Return** the public variables = (G_{pub}, n, k, d, p) and the private variables = (H, P, S)

Table 2 Encryption algorithm

Algorithm 2 Encryption
Input: The public variables = (G_{pub}, n, k, d, p) and Message M
Output: The ciphertext C
1: Convert M to MEC block message M_π of length k Convert M to M_2 Convert M_2 to $M_{\text{number of } L_m}$ by Horner algorithm Map $M_{\text{number of } L_m}$ to M_π
2: Random error vector E of length n
3: $C_\pi = M_\pi G_{pub} + E$
4: **Return** the ciphertext C

scramble a matrix of dimension k x k and apply a Gaussian elimination method to check its invertibility. For the construction of the permutation matrix, we rotate a row of an n x n identity matrix over Gaussian integer that contains a single element from the set {1, -1, i, -i}.

The public key is obtained as $G_{pub} = S^{-1}GP \bmod \pi$ for the encoding process. The generator matrix is randomized by secret scrambles and permutation matrices to make its structure non-systematic. Since no one knows the secret parameters of the public key, it is difficult to perform the correct decryption. Hence, the public key can be published to everyone, not only the sender. The key generation algorithm is described in Table 1.

2.2.2 Encryption

In the encryption procedure, the system first converts an original message from the sender to a k-length symbol message as MEC block codes. The binary vector of the plaintext is transformed to a corresponding representation vector in base number of message elements (L_m) using the Horner algorithm [19]. Then the system maps this vector to the primitive elements of MEC codes. For example, in the case of MEC codes M_π that π=2+i, primitive element L = {1, i, -1, -i} and L_m ={0, 1, -1, i, -i, 1+i, 1-i, -1+i, -1-i}. With 9 possible elements of L_m, the binary message vector M_2 = (0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0) is transformed to M_9 = (1, 0, 5, 0) and is mapped to M_{2+i} = (1, 0, 1+i, 0) with members in L_m, respectively.

In the next step, the public key is used to encrypt the plaintext to ciphertext. The system generates an arbitrary non-zero error vector E of Mannheim weight. This means the error values (±1, ±i) will be added at random positions l ($0 \leq l \leq n-1$). Finally, the system produces the syndrome of each cipher vector with different generated error vectors. The encryption algorithm is described in Table 2.

2.2.3 Decryption

The decryption algorithm is described in Table 3. Each private parameter in the private key is used to decrypt the ciphertext to the original message. First, the received ciphertext C is multiplied by the inverse of the permutation matrix to remove permutations in the cipher vector. To decode the cipher vector, the parity check matrix is used to compute the error values and their locations occurring in the syndrome. After decoding, the cipher vector must remove the scrambles. The scramble matrix is multiplied by the decoded cipher vector and then the plaintext is given as MEC block codes. In order to obtain the original message, the system converts MEC codes M_π to M using reversion of the transformation process in the encryption procedure. The MEC codes are mapped to the corresponding representation vector of L_m. Then, this vector is transformed into a binary vector using the Horner algorithm. Finally, the binary vector is reconstructed to the desired message.

3. Results and discussion

3.1 Computational assessment

In this section, we examined computation of the proposed cryptosystem. The computational assessment is divided into two sections: 1) theoretical computations, and 2) empirical computations.

3.1.1 Theoretical computation

In theoretical computations, we estimate the computational complexity cost of key generation (Com_K), encryption (Com_E) and decryption (Com_D) with regard to the

Table 3 Decryption algorithm

Algorithm 3 Decryption
INPUT: The private variables = (H, P, S) and The ciphertext C
OUTPUT: The message M
1: $Y = C_\pi P^{-1}$
2: Y' = Decode(H, Y)
3: M_π = Y'S mod π
4: Convert M_π to original message M
Map M_π to $M_{\text{number of } L_m}$
Convert $M_{\text{number of } L_m}$ to M_2 by Horner algorithm
Convert M_2 to M
5: **Return** the message M

Table 4 Comparison of the computational complexity costs of key generation, encryption and decryption of the proposed cryptosystem and original cryptosystem

Cryptosystem	**Original McEliece1 [2]**	**Original McEliece2 [20]**	**Gaussian McEliece**
Key generation	$O(n^3)$	$O(t^3(n-k)) + O((n-k)^3) + O(k^2n + n^2)$	$O(p(n-k)) + O(n) + O(n-k) + O(n) + O(k^2) + O(n^3) + O(k^3)$
Encryption	$O(n^2)$	$O(kn) + O(t)$	$O(k) + O(kn) + O(n-k)$
Decryption	$O(n^2)$	$O(ntm^2) + O(n) + O(k^2)$	$O(n^2) + O(p) + O(k^2) + O(k)$

complexity of additions, multiplications, and modular operations. Following the key generation algorithm in Table 1, the MEC codes are constructed over G_{π^r}. Then, the private and public variables are generated for further processes. Thus, the key generation complexity is computed as Eq. (3):

$$Com_K = Com_{con} + Com_{pri} + Com_{pub} \quad (3)$$

where $Com_{con} = O(p(n-k))$ is the complexity of MEC code construction over G_{π^r}. The complexity of private variables is $Com_{pri} = O(n) + O(n-k) + O(n) + O(k^2) + O(n^3) + O(k^3)$ consisting of the construction complexity of the parity check matrix H, the generator matrix G, the permutation matrix P, the scramble matrix S, as well as inversions of P and S. Public key construction is the only public variable that is important in the complexity. The complexity of public variables is $Com_{pub} = O(k^2n) + O(n^2k)$.

In encryption, the original message is converted to the finite Gaussian integer field M_π. Then, the ciphertext is generated by multiplying Mπ with the public key Gpub and adding a random error vector E. The complexity cost of encryption can be computed as Eq. (4):

$$Com_E = Com_c + Com_m(M_\pi G_{pub}) + Com_a(E) \quad (4)$$

where $Com_c = O(k)$ is the complexity of the k-symbol conversion process. $Com_m(M_\pi G_{pub}) = O(kn)$ is a complexity of multiplying the message and public key, and the final part is the complexity of adding the error to ciphertext as $Com_a(E) = O(n-k)$.

The decryption algorithm is shown in Table 3. The received ciphertext is decrypted by multiplying the inverse of P. Then, the decoding process corrects error vectors and removes the scramble by multiplication of the scramble matrix. Finally, the message in the Gaussian integer field is transformed to the desired original message. The complexity cost of decryption can be computed as Eq. (5):

$$Com_D = Com_m(C_\pi P^{-1}) + Com_{de}(Y) + Com_m(Y'S) + Com_c \quad (5)$$

where $Com_m(C_\pi P^{-1}) = O(n^2)$ is the complexity of multiplying the ciphertext by the inverse of P. The decoding complexity is computed as $Com_{de}(Y) = O(p)$. Also, $Com_m(Y'S) = O(k^2)$ is the complexity of multiplying the k-symbol matrix by the inverse of S. Finally, the complexity of the k-symbol conversion process, Com_c, has the same meaning as in Eq. (4).

Comparison of the theoretical computation of the processes and the computational complexity from the initial parameters of the original cryptosystem [2] were roughly estimated. The results of [20] are presented in Table 4. The theoretical results show that the complexity cost of the key generation process is the highest among the three main processes. The scramble matrix, permutation matrix, and their inversion execution are the most time-consuming steps. As shown in Table 4, there is $O(k^2n + n^2)$ for the original McEliece2 and $O(n) + O(k^2) + O(n^3) + O(k^3)$ for the Gaussian McEliece. Moreover, in the proposed algorithm, the disadvantage is the large complexity cost of multiplying ciphertext with the inverse of P, that requires $O(n^2)$ of time. Subsequently, the optimized algorithm is presented further to provide a satisfactory computational complexity cost of the decryption process.

3.1.2 Empirical computation

For empirical computations, we implemented the proposed algorithm and calculate the runtime using a personal computer with the following environment: CPU Intel Core i5 3210M 2.5 GHz., RAM 4 GB, OS Windows 10 64 bit, using Python 3.5 as the programing language. In our implementation, we use some specific libraries of Python, such as Numpy and Sympy, in the calculation processes of the cryptosystem. The elements and the primitive polynomial elements of MEC codes are performed by the polynomial class with coefficients in the form of Gaussian integers. The block codes along with other variables for encryption and decryption are represented as matrices with their basic addition and multiplication operations. The default modular operation is also overridden for convenience.

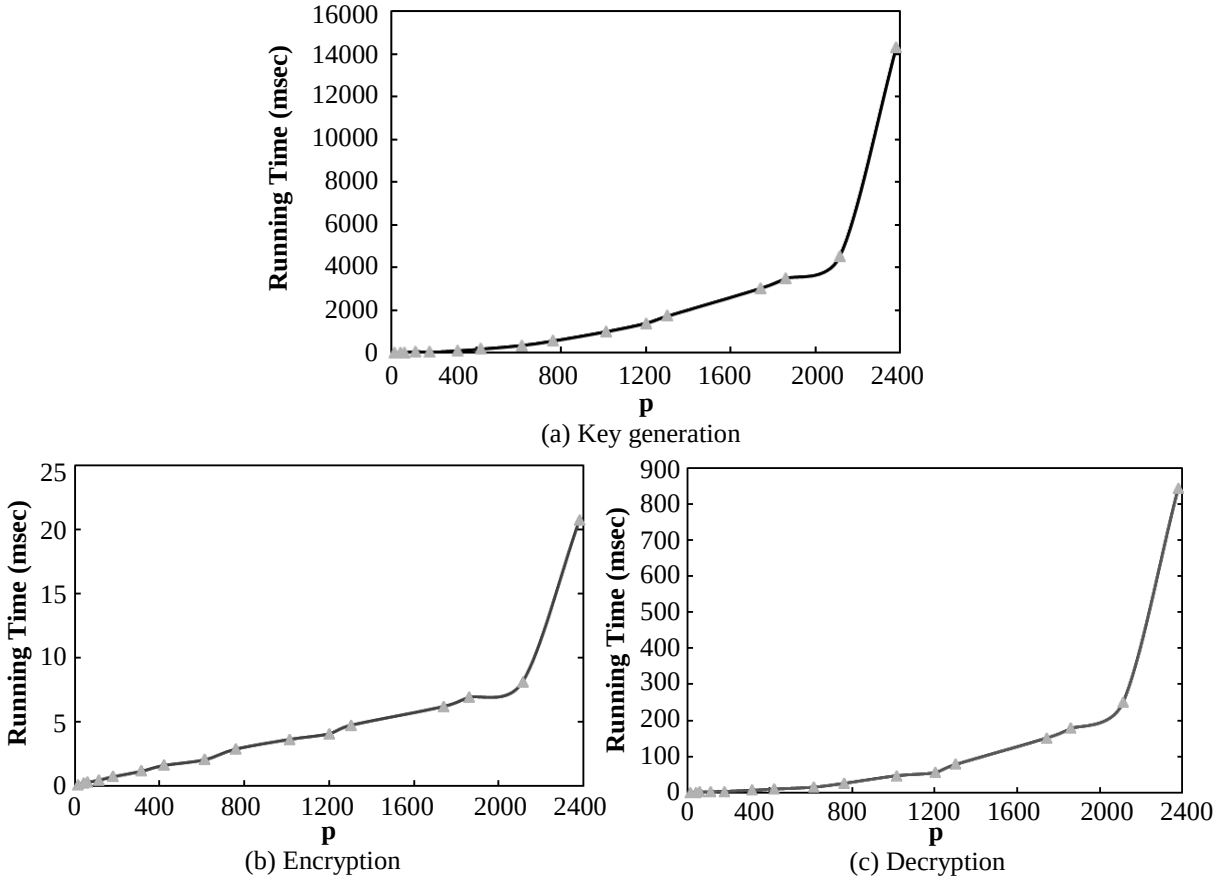

Figure 2 The average runtime of (a) key generation, (b) encryption and (c) decryption with varying values of p

Table 5 Size of keys for the value of p between 421 and 2381

p	421	1013	1861	2381
Public key	12 kB	79 kB	296 kB	529 kB
Private key	0.12 kB	0.32 kB	0.64 kB	0.89 kB

To calculate runtime, we use a performance counter that provides the time in fractional seconds for the three main processes. First, the average runtime of key generation is calculated from one hundred trials of random code construction and key pair generation using the same initial parameters. This process includes all pre-computations such as inversion of P, symbol matching table, etc. Next, the average runtime of encryption and decryption are measured from encoding and decoding a thousand random messages with the same keys.

Various values of p are used to increase the security level of the cryptosystem. In Figure 2 (a), the key generation steps with p = 421, 1013, 1861, 2381 indicate average times of about 0.19, 1.01, 3.51 and 14.34 seconds, respectively. The average encryption times are about 1.63, 3.66, 6.94 and 20.76 milliseconds, and decryption times are about 9.32, 46.26, 177.44 and 842.32 milliseconds for the same parameters p summarized in Figure 2 (b) and (c). This result shows that if we choose larger values of p to obtain higher security, the cryptosystem provides longer encoding and decoding times.

The private and public key sizes are measured as follows: 1) the public key matrix G_{pub} requires a kn-symbol length and has p elements per symbol of the finite field G_{π}. In this case, the required memory for storing the public key G_{pub} is approximately equal to $k \cdot [n \cdot \log_2 p]$ bits. 2) the private key H requires an n-symbol length with the same field. Therefore, the required memory to store the private key is approximately equal to $n \cdot \log_2 p$ bits. In Table 5, we compute the private and public key sizes with p = 421, 1013, 1861, 2381. The key pair size increases corresponding to the higher value of p. This shows the trade-off between the level of security and the key length.

To compare the empirical computation of the processes, we summarize the existing implementations in different environments of the original and proposed cryptosystems in Table 6. Strenzke [21] presented the implementation of a McEliece cryptosystem with n equals 1024 and a security level approximately equal to 60 (estimated from the work factor of an ISD attack) on a smart card with two different platforms. For an embedded device, the encryption and decryption processes took 1260 ms and 980 ms, respectively on an Infineon SLE 76 chip. For a personal computer, Intel Core Duo with Linux OS, the encryption and decryption processes took 0.75 ms and 0.8 ms, respectively. The key generation process was not featured in the implementation because of the card's RAM size issue. Takuya et al. [22] implemented a cryptosystem on a personal computer with AMD Phenom CPU. They reported timing for parameter n equals 2048 and a security level approximately equal to 80. The key generation, encryption and decryption times are 29.85 s, 1.06 ms, and 116.41 ms, respectively. For comparison, the proposed cryptosystem, with the same security, was implemented on an Intel Core i5.

Table 6 Comparison of the computational time of key generation, encryption and decryption of the proposed and original cryptosystems

Cryptosystem	Key generation	Encryption	Decryption	Environment
Goppa McEliece [21] n = 1024 Security level ~60	no feature	1260ms	980ms	Infineon SLE76CF5120P controller 33 MHz
Goppa McEliece [21] n = 1024 Security level ~60	no feature	0.75ms	0.8ms	Intel core Duo T7300 2GHz, Linux kernel v.2.3.2.4, GCC-4.1.3 compiler
Goppa McEliece [22] n = 2048 Security level ~80	29.85s	1.06ms	116.41ms	AMD Phenom II 1090T 3.2GHz, Window 7, Visual C++ 2010 compiler
Gaussian McEliece n = 1013 Security level ~60	1.01s	3.66ms	77.58ms	Intel Core i5 3210M 2.5 GHz, Windows 10, Python 3.5 complier

Table 7 Comparison of the number of trials for scramble and permutation matrices and public key size of the proposed and original cryptosystems

Cryptosystem	Original McEliece	Gaussian McEliece		
$(n, k)_p$	(1024, 524)	$(105,103)_{421}$	$(253, 251)_{1013}$	$(465,463)_{1861}$
Public key size	67 kB	12kB	79kB	296kB
N_S	$\approx 2^{275099}$	$\approx 2^{57757}$	$\approx 2^{421139}$	$\approx 2^{1427306}$
N_P	$\approx 2^{8770}$	$\approx 2^{547}$	$\approx 2^{1646}$	$\approx 2^{3440}$
N_{Total}	$\approx 2^{283869}$	$\approx 2^{58304}$	$\approx 2^{422785}$	$\approx 2^{1430746}$

The key generation, encryption and decryption processes took 1.01 s, 3.66 ms and 77.58 ms, respectively. Although the computation time in Gaussian McEliece is rather long in the encryption and decryption processes, the key generation is an efficient process for modern cryptosystems using a short-term key.

3.2 Security assessment

In this section, we discuss the security of proposed cryptosystem against two types of practical attacks. The security assessment is divided into two sections: key recovering attack and decoding attack. The former tries to randomly guess the secret matrices for recovering the proper private key. The latter tries to decode the ciphertext to an original message without knowledge of the private key.

3.2.1 Key recovering attack

A key recovering attack aims to recover the private key from public variables of the cryptosystem. A brute force attack is one in which the adversary tries to randomly search for a probable key until the actual key is detected. In a brute force attack, we assess the security of the proposed cryptosystem by estimating the complexity of random guessing of two secret matrices of the private key, the scrambled and permutation matrices. First, the scrambled matrix is generated by selecting an element from G_π of order p. The matrix has dimension k x k and is non-singular, thus the rows of the matrix must be linearly independent. Therefore, the number of trials for the scrambled matrix is given as Eq. (6). For further simplification, we use the following inequality in [10] as Eq. (7). Additionally, the permutation matrix has dimensions n x n and can be selected as a single element from the set {1, -1, i, -i}. Thus, the number of possible permutation matrices is 4n!. For the factorial function, we use Stirling's approximation [23] yielding a more precise formula. Therefore, the number of trials for the permutation matrix is given as Eq. (8).

$$N_S = \prod_{i=0}^{k-1}\left(p^k - p^i\right) \tag{6}$$

$$N_S \le p^{k(k+1)-1} \tag{7}$$

$$N_p \approx 4\sqrt{2\pi n}\, n^n e^{-n} \tag{8}$$

In Table 7, we compare the number of trials for scrambled and permutation matrices and public key size to the original McEliece cryptosystem. In order to gain computational performance, we use the optimal number of elements in the finite field to construct the scrambled matrix. As indicated in the table, for G_π with p = 1013, the proposed cryptosystem can provide a massive increase in the total number of trials N_{total} with a bit larger key size compared to the original cryptosystem. Hence, the number of trials is increased by a vast increment and can reach a maximum number according to the suitable hardware in various aspects without extending the key size.

3.2.2 Decoding attack

In a decoding attack, the adversary attempts to decode the ciphertext without knowing the private key. An exhaustive decoding attack is a brute force approach based on comparison of the received ciphertext C_π with each generated ciphertext C_{gen} that is defined by multiplying all possible messages to the public key. In order to find the closest generated ciphertext, we determine the distance between each C_{gen} and C_π which is less than Mannheim distance d_M of code. Thus, the work factor of exhaustive decoding as WF_{ED} is obtained as Eq. (9).

$$WF_{ED} = 2^{n \log_2 p} \tag{9}$$

A syndrome decoding attack is an efficient approach introduced by Jochemsz [24]. In this attack, the cryptanalyst calculates the parity matrix H_{pub} corresponding to G_{pub}. $G_{pub}\cdot(H_{pub})^T = 0$, which is the right syndrome of the transmitted ciphertext C_π is computed as $C_\pi\cdot(H_{pub})^T =$

Table 8 The work factor of decoding attack of proposed cryptosystem and original cryptosystem

Cryptosystem	Original McEliece	Gaussian McEliece		
(n, k)	(1024, 524)	$(105,103)_{421}$	$(253, 251)_{1013}$	$(465,463)_{1861}$
Public key size	67 kB	12kB	79kB	296kB
WF_{ED}	2^{524}	2^{945}	2^{2530}	2^{5115}
WF_{SD}	$\approx 2^{284}$	$\approx 2^{125}$	$\approx 2^{164}$	$\approx 2^{201}$
WF_{ISD}	$\approx 2^{64.2}$	$\approx 2^{55.6}$	$\approx 2^{62.6}$	$\approx 2^{68.6}$

$(M_{\pi}G_{pub})(H_{pub})^T + E(H_{pub})^T = E(H_{pub})^T$. The attacker tries to generate all possible error vectors of length n with the weight $t \leq d_M$. Therefore, the work factor of trails of the possible error vectors and success in achieving the correct syndrome as WF_{SD} is obtained as Eq. (10):

$$WF_{SD} = \sum_{i=0}^{t} \binom{n \log_2 p}{i \log_2 p} \quad (10)$$

Another type of decoding attack is an information set decoding attack (ISD). It represents a powerful method to search the error vector in ciphertext without exploiting the secret code structure. The idea is to use restriction of generator matrix to find a set of information vectors that contain no errors in any position. Then, the ciphertext is multiplied by the inverse of submatrix to decrypt the original plaintext. In this paper, we use the original algorithm introduced by Prange [25] to analyze the complexity of the proposed cryptosystem against an ISD attack. By Prange's approach, the work factor of an ISD attack as WF_{ISD} is obtained as Eq. (11).

$$WF_{ISD} \approx k^3 \left(1-\frac{t}{n}\right)^{-k} \quad (11)$$

In Table 8, we calculate the work factor of all decoding attacks of both the original and the proposed cryptosystems. The results of Table 8 show that the wide range of code over Gaussian integer provides enormous workload for an exhaustive decoding attack. With p=421, the smallest key size parameter, we can increase the work factor from 2^{524} to 2^{945} compared to the original cryptosystem. However, in cryptanalyzed decoding attacks, the increase of code range yields less benefit than the exhaustive method. The proposed cryptosystem has a lower work factor in a syndrome decoding attack, but a larger work factor in the information set decoding attack compared to the original cryptography. For instance, with p=1861 we obtain a 2^{201} workload for a syndrome decoding attack and a $2^{68.6}$ workload for an information set decoding attack. This shows a limitation of codes when the expanded range of code syndrome is not efficient enough.

4. Conclusions

This paper enhanced a very strong post-quantum code-based cryptography the McEliece asymmetric key cryptosystem using Gaussian integers. We show that larger keyspace and error syndrome ranges seem to offer more security against ordinary attacks compared to a traditional cryptosystem. Furthermore, there is an interesting observation from implementation our work. The optimal parameters and the trade-off between the strength of a cryptosystem and its performance are seen. The limitation of the proposed cryptosystem is the greater field of original code over Gaussian integers, which can slightly increase the work factor of particular decoding attacks. In the future, the specific structure of the code might be applied in such a way to obtain better security efficiency of cryptographic schemes.

5. References

[1] Shor PW. Algorithms for quantum computation: discrete logarithms and factoring. Proceedings of 35th Annual Symposium on Foundations of Computer Science; 1994 Nov 20-22; Santa Fe, USA. USA: IEEE; 1994. p. 124-34.

[2] Mceliece RJ. A public-key cryptosystem based on algebraic coding theory. JPL DSN Progress Report 1978;44:114-6.

[3] Niederreiter H. Knapsack type cryptosystems and algebraic coding theory. Probl Contr Inform Theor. 1986;15:159-66.

[4] Sidelnikov VM, Shestakov SO. On insecurity of cryptosystems based on generalized Reed-Solomon codes. Discrete Math Appl. 1992;2:439-44.

[5] Sidelnikov VM. A public-key cryptosystem based on binary Reed-Muller codes. Discrete Math Appl. 1994; 4:191-208.

[6] Minder L, Shokrollahi A. Cryptanalysis of the Sidelnikov cryptosystem. In: Naor M, editor. Advances in Cryptology – EUROCRYPT 2007; 2007 May 20-24; Barcelona, Spain. Berlin: Springer; 2007. p. 347-60.

[7] Monico C, Rosenthal J, Shokrollahi A. Using low density parity check codes in the McEliece cryptosystem. 2000 IEEE International Symposium on Information Theory; 2000 Jun 25-30; Sorrento, Italy. USA: IEEE; 2000. p. 215.

[8] Baldi M, Bodrato M, Chiaraluce F. A new analysis of the McEliece cryptosystem based on QC-LDPC codes. In: Ostrovsky R, De Prisco R, Visconti I, editors. Security and Cryptography for Networks 2008; 2008 Sep 10-12; Amalfi, Italy. Berlin: Springer; 2008. p. 246-62.

[9] Misoczki R, Tillich JP, Sendrier N, Barreto PSLM. MDPC-McEliece: new McEliece variants from moderate density parity-check codes. 2013 IEEE International Symposium on Information Theory; 2013 Jul 7-12; Istanbul, Turkey. USA: IEEE; 2013. p. 2069-73.

[10] Shrestha SR, Kim YS. New McEliece cryptosystem based on polar codes as a candidate for post-quantum cryptography. 2014 14th International Symposium on Communications and Information Technologies; 2014 Sep 24-26; Incheon, South Korea. USA: IEEE; 2014. p. 368-72.

[11] Hooshmand R, Shooshtari MK, Eghlidos T, Aref MR. Reducing the key length of mceliece cryptosystem using polar codes. 11th International ISC Conference on Information Security and Cryptology; 2014 Sep 3-4; Tehran, Iran. USA: IEEE; 2014. p. 104-8.

[12] Bardet M, Chaulet J, Dragoi V, Otmani A, Tillich JP. Cryptanalysis of the McEliece public key cryptosystem based on polar codes. In: Takagi T,

editor. Post-Quantum Cryptography 2016; 2006 Feb 24-26; Fukuoka, Japan. Berlin: Springer; 2016. p. 118-43.

[13] Hooshmand R, Shooshtari MK, Aref MR. PKC-PC: a variant of the McEliece public key cryptosystem based on polar codes. arXiv:1712.07672. 2017:1-11.

[14] Pradhan S, Sharma BK. A modified variant of RSA algorithm for Gaussian integers. Int J Inform Netw Secur. 2013;2:322-6.

[15] Mohamed E, Elkamchouchi H. Elliptic curve cryptography over Gaussian integers. Int J Comput Sci Netw Secur. 2009;9:413-6.

[16] Nanda AK, Nayak R, Awasthi LK. NTRU with Gaussian integer matrix. Int J Adv Res Comput Sci Software Eng. 2015;5:359-65.

[17] Huber K. Codes over Gaussian integers. IEEE Trans Inform Theor. 1994;40:207-16.

[18] Huber K. Decoding algorithms for block codes over two-dimensional signal-constellations. IEEE International Symposium on Information Theory; 1994 Jun 27 - Jul 1; Trondheim, Norway. USA: IEEE; 1994. p. 472.

[19] Freudenberger J, Ghaboussi F, Shavgulidze S. New coding techniques for codes over Gaussian integers. IEEE Trans Comm. 2013;61:3114-24.

[20] Engelbert D, Overbeck R, Schmidt A. A summary of McEliece-Type cryptosystems and their security. J Math Cryptol. 2007;1(2):151-99.

[21] Strenzke F. A smart card implementation of the McEliece PKC. In: Samarati P, Tunstall M, Posegga J, Markantonakis K, Sauveron D, editors. Information Security Theory and Practices Security and Privacy of Pervasive Systems and Smart Devices; 2010 Apr 12-14; Passau, Germany. Berlin: Springer; 2010. p. 47-59.

[22] Takuya S, Kirill M, Tsuyoshi T. Efficient implementation of the McEliece cryptosystem. Computer Security Symposium 2011; 2011 Oct 19-21; Niigata, Japan. Japan: Information Processing Society of Japan; 2011. p. 582-7.

[23] Scheinerman EA. Collections. Mathematics: a discrete introduction. Boston: Cengage Learning; 2012. p. 40-3.

[24] Jochemsz E. Goppa codes & the McEliece cryptosystem [dissertation]. Amsterdam, Netherlands: VU University; 2002.

[25] Prange E. The use of information sets in decoding cyclic codes. IRE Trans Inform Theor. 1962;8:5-9.