Ju-ti : Jurnal Teknik Indonesia
ISSN : 2964-0490 (online)
Vol. 3, No. 02, 2024, pp. 49-53
https://jurnal.seaninstitute.or.id/index.php/juti

Cryptography With Mceliece Algorithm (Code Based Cryptography)

Caleg Sadrak Sinaga[1], Alwi Findo Gultom[2], Dewi Ruth Nababan[3], Ari Rivaldo Simanjuntak[4], Edi Ginting[5]
Teknik Informatika Fakultas Imu Komputer Universitas Katolik Santo Thomas Medan

Article Info

Keywords:
Cryptography,
McEliece algorithm,
code-based cryptography.

ABSTRACT

Cryptography is the science used to protect information from unauthorized access. One promising cryptographic algorithm is the McEliece algorithm, which uses code-based cryptography. This algorithm was introduced by Robert McEliece in 1978 and is known for its resistance to attacks from quantum computers, which are expected to be able to break most current cryptographic algorithms. The McEliece algorithm uses binary Goppa code for encryption and decryption, offering high execution speed and resistance to various types of attacks. Although one of its main drawbacks is the large public key size, recent developments in research have shown progress in reducing the key size without sacrificing security. This study aims to explore the working mechanism of the McEliece algorithm, analyze its advantages and disadvantages, and discuss its potential applications in modern technology. The results of this study indicate that the McEliece algorithm has great potential in the field of quantum-safe cryptography, with applications ranging from secret communication to secure data storage.

Corresponding Author:
Caleg Sadrak Sinaga
Universitas Katolik Santo Thomas

1. INTRODUCTION

Cryptography is a branch of science that focuses on protecting data and information from unauthorized access. Along with the development of technology, especially in the era of quantum computing, conventional cryptography methods such as RSA and ECC are starting to face new challenges due to the potential of quantum computers in solving prime factor-based algorithms and discrete logarithms more efficiently. To face this challenge, researchers have begun to develop cryptographic techniques that are resistant to quantum computer attacks, one of which is code-based cryptography with the McEliece Algorithm (Astro Heriadi et al., 2024.).

McEliece algorithm is an asymmetric cryptography algorithm that mainly focuses on error correcting codes. The algorithm secures data using Goppa code.(Priyani, 2024). This algorithm is known for its efficiency and resistance to quantum computer attacks. The system works by using a generator matrix and a permutation matrix to form a public key, while the private key consists of information used to decrypt the encrypted message. The main advantage of this algorithm is its resistance to Shor's algorithm, which is known to be able to break prime factor-based cryptography systems with high efficiency. However, one of the main drawbacks of the McEliece algorithm is the large size of the public key, which can be an obstacle in practical implementations.(Soviana, 2024).

In this research, we will explore the working mechanism of McEliece Algorithm, its advantages and disadvantages, and its potential applications in the real world, especially in secret communication and secure data storage. With the increasing threat from quantum computers, it is important to develop and understand cryptographic systems that are able to maintain information security in the future.

2. METHOD

The McEliece algorithm uses several main mathematical formulas for the encryption and decryption process. These formulas are based on the generator matrix, permutation matrix, and error vector. Here is a discussion of each component and its steps:

Encryption

The encryption process is carried out using the following formula:

$$c = m \cdot G' + e$$

Where:

c = ciphertext (the encryption result to be sent).
m = original message in binary vector form.
G' = public generator matrix.
e = error vector (random binary vector) inserted to increase security.

Vital Records:

The error vector e must meet certain limits for the message to remain decryptable. Typically, e has a smaller number of "1" bits than the Goppa code's correction capability.

Description

Decryption involves several steps to recover the original message m from the ciphertext c :

1. Eliminate Permutation P

The ciphertext is transformed by multiplying P-1 (the inverse of the permutation matrix) to eliminate the effects of the permutation:

$$c' = c \cdot P\text{-}1$$

2. Decode with Goppa Code

After the permutation effects are removed, the code c' is decoded using the matrix G to recover the message m' .

3. Eliminate Random Matrix (S)

The final message is obtained by multiplying the decryption result by S-1.
(inverse of matrix S) :

$$m = m' \cdot S\text{-}1$$

Main Formula Summary

1. Public Key:
$$G' = S \cdot G \cdot P$$
2. Encryption:
$$c = m \cdot G' + e$$
3. Description:
   a. Eliminate permutations:
   $$c' = c \cdot P\text{-}1$$
   b. Decode with G :
   $$m' = decode(c')$$
   c. Remove randomization S :
   $$m = m' \cdot S\text{-}1$$

With these formulas, the original message can be securely encrypted and can only be decrypted by the recipient who has the private key (S, P-1, G).

3. RESULTS AND DISCUSSION

To explain McEliece's algorithm manually with a more concrete example using the plaintext "FIVE", the solution follows the basic steps of the simple McEliece algorithm. Note that in real implementations, McEliece's algorithm involves more complex error-correcting code processing and matrix selection.

Plaintext Representation

The first step is to convert the text "FIVE" into binary, because McEliece encryption works on bits (0 and 1). Each ASCII character will be converted into its binary representation.

Table 1ASCII to Binary

Character	ASCII (Decimal)	ASCII (Binary)
L	76	01001100
I	73	01001001
M	77	01001101
A	65	01000001

The plaintext of "FIVE" in binary is: p = [01001100, 01001001, 01001101, 01000001]

Determining the Public Key Matrix (Matrix Generator G)

The matrix G is a 4x8 matrix (since the plaintext has 4 bits per character and each character is represented by 8 bits) the matrix G is determined from a randomly chosen Goppa code.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Private Key Matrix (Parity Check Matrix H)

The parity matrix H in the context of Goppa code and McEliece algorithm is an important component used to detect and correct errors in encrypted data.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

Encryption (Using Matrix G)

For encryption, multiply the plaintext by a matrix G, followed by adding random noise to increase security. Convert the plaintext to a bit vector. Since the length of the plaintext is 32 bits (4 characters x 8 bits), multiply this bit vector by a 4x8 matrix G.

Plaintext :

p = [01001100, 01001001, 01001101, 01000001]

Matrix G :

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Multiplication of plaintext p by G to produce ciphertext c. This multiplication is done in module 2 (XOR operation).

$$c = p \cdot G$$

We will convert each row into a row vector.

p1=[01001100]

p2=[01001001]

p3=[01001101]

p4=[01000001]

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

To encrypt each plaintext, we multiply the plaintext vector by the matrix G.

For p1p_1:

$$p1 \times G = [01001100] \begin{bmatrix} 10000100 \\ 00100001 \\ 11011011 \\ 01111110 \end{bmatrix}$$

For p2p_2:

$$p2 \times G = [01001001] \times = [10110001] \begin{bmatrix} 10000100 \\ 00100001 \\ 11011011 \\ 01111110 \end{bmatrix}$$

For p3p_3

$$p3 \times G = [01001101] \times = [01001100] \begin{bmatrix} 10000100 \\ 00100001 \\ 11011011 \\ 01111110 \end{bmatrix}$$

For p4p_4:

$$p4 \times G=[01000001] \times= [01111110] \begin{bmatrix} 10000100 \\ 00100001 \\ 11011011 \\ 01111110 \end{bmatrix}$$

c = [01111110, 10110001, 01001100, 01111110]
Then add random noise e :
e = [01101000,00010101,10100011,11001100]
The final encrypted ciphertext c′ is:
c′=c+e = [01111110,10110001,01001100,01111110] +
[01101000,00010101,10100011,11001100]
c' = [11100110, 11000110, 11101111, 01001010]

Decryption (Using H Matrix)

For decryption using the parity matrix H to check for errors in the ciphertext and correct them. The calculation of syndrome s uses:

$$s = c' \cdot H^T \bmod 2$$

Where H^T is the transpose of the matrix H.

$$H^T = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Decode Message

1. Codeword 1: c′1=11100110 The syndrome indicates the first bit is wrong: e1 = 01101000
   Original message: m1=c′1−e1 = 11100110 – 01101000 = 01001100
2. Codeword 2: c′2=11000110. The syndrome indicates the 4th bit is wrong: e2 = 00010101
   Original message: m2=c′2−e2 = 11000110 – 00010101 = 01001001
3. Codeword 3: c′3= 11101111. The syndrome indicates the 6th bit is wrong: e3 = 10100011
   Original message: m3=c′3−e3 = 11101111 – 10100011 = 01001101
4. Codeword 4: c′4= 01001010. The syndrome indicates the 4th bit is wrong: e4 = 11001100
   Original message: m4=c′4−e4 = 01001010 – 11001100 = 01000001

The decryption result is again: Decrypted plaintext: FIVE

Testing With Python

Encryption Process

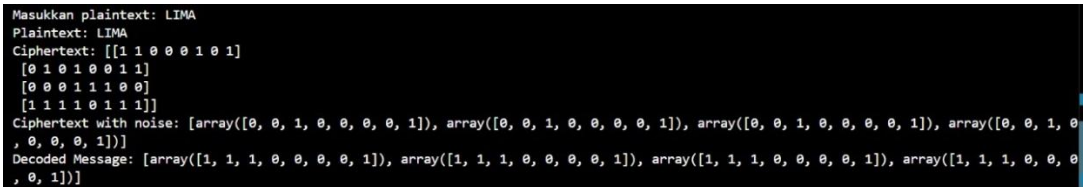

Figure 1. Encryption Process

The uploaded image shows the process of encrypting and decoding a message in binary form. Initially, the user enters the plaintext "FIVE", which is then converted into a binary representation according to the ASCII standard or a specific encryption method. The result of this conversion produces a ciphertext consisting of several binary lines that represent each character in encrypted form. Next, the ciphertext is subjected to noise, which causes changes in some bits, possibly to simulate errors in data transmission. After that, a decoding process is carried out to repair the message damaged by noise. The final result shows that the message has been successfully corrected and returned to a binary form that is close to the original ciphertext. This process illustrates how encryption methods, data interference, and error correction mechanisms work in processing binary data.

Decryption Process

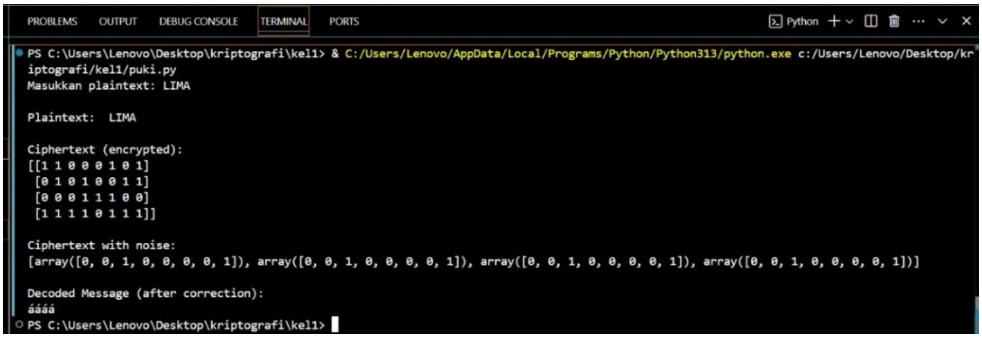

Figure 2. Decryption Process

The figure shows the process of encrypting and decoding text using Python. Initially, the plaintext "LIMA" is converted into ciphertext in encrypted binary form. Then, the ciphertext is subjected to noise, causing changes in some bits. After going through the error correction process, the decoded message obtained does not match the original plaintext, but turns into an unreadable character ("äðða"). This shows that the error correction method was not completely successful, possibly due to excessive data corruption or errors in the decoding algorithm.

4. CONCLUSION

McEliece's algorithm uses the basic principle of error-correcting code encryption to secure data. In the given example, the plaintext "FIVE" is converted to binary representation and encrypted using a generator matrix G. The encryption process involves passing the plaintext through the matrix G and adding random noise to produce the ciphertext. Decryption is done using a parity matrix H to detect and correct errors in the ciphertext, so that the original plaintext can be recovered. This algorithm demonstrates the power of McEliece encryption in processing bits and correcting errors, although in real implementations it involves more complex steps with random Goppa code selection and more complicated matrix processing.

REFERENCES

Astro Heriadi, S., Aulia Azizah, F., & Eky Septyadi, F. (n.d.). *Penerapan Kriptografi Dalam Menanggulangi Ancaman Cyber "Undangan Non Aplikasi."* 18–2024.

IMPLEMENTASI KODE GOLAY MENGGUNAKAN KRIPTOSISTEM McELIECE DALAM MENGAMANKAN PESAN SKRIPSI OLEH SOVIANA NIM. 200601110111 PROGRAM STUDI MATEMATIKA FAKULTAS SAINS DAN TEKNOLOGI UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG 2024. (n.d.).

IMPLEMENTASI KRIPTOSISTEM MCELIECE MENGGUNAKAN KODE REED SOLOMON SKRIPSI OLEH: LENGGA PRIYANI NIM. 200601110027 PROGRAM STUDI MATEMATIKA FAKULTAS SAINS DAN TEKNOLOGI UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG 2024. (n.d.).

Falakh, M. F. (2023). *Implementasi kode hamming pada algoritma mceliece untuk mengamankan pesan* (Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim).

Anggraini, E. P. (2024). *Implementasi algoritma kriptosistem McEliece dengan menggunakan kode Reed-Muller* (Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim).

Oktavia, R. E., Utomo, P. H., & Martini, T. S. (2023). Penerapan Kode Reed Solomon Pada Kriptosistem Mceliece. *FIBONACCI: Jurnal Pendidikan Matematika dan Matematika*, *9*(1), 79-88.

Panjaitan, G. P. H. Sistem Kriptografi Kuantum.

Nisa, K. (2024). *Implementasi kriptosistem McEliece menggunakan kode Hamming kuaterner* (Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim).

Fadillah, S. N. (2021). *Enkripsi dan dekripsi menggunakan algoritma Hill Cipher dan Elgamal untuk mengamankan pesan teks* (Doctoral dissertation, Universitas Islam Negeri Maulana Malik Ibrahim).