International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:9, No:10, 2015

Comparison between Separable and Irreducible Goppa Code in McEliece Cryptosystem

Thuraya M. Qaradaghi, Newroz N. Abdulrazaq

***Abstract*—**The McEliece cryptosystem is an asymmetric type of cryptography based on error correction code. The classical McEliece used irreducible binary Goppa code which considered unbreakable until now especially with parameter [1024, 524, and 101], but it is suffering from large public key matrix which leads to be difficult to be used practically. In this work Irreducible and Separable Goppa codes have been introduced. The Irreducible and Separable Goppa codes used are with flexible parameters and dynamic error vectors. A Comparison between Separable and Irreducible Goppa code in McEliece Cryptosystem has been done. For encryption stage, to get better result for comparison, two types of testing have been chosen; in the first one the random message is constant while the parameters of Goppa code have been changed. But for the second test, the parameters of Goppa code are constant (m=8 and t=10) while the random message have been changed. The results show that the time needed to calculate parity check matrix in separable are higher than the one for irreducible McEliece cryptosystem, which is considered expected results due to calculate extra parity check matrix in decryption process for $g^2(z)$ in separable type, and the time needed to execute error locator in decryption stage in separable type is better than the time needed to calculate it in irreducible type. The proposed implementation has been done by Visual studio C#.

***Keywords*—**McEliece cryptosystem, Goppa code, separable, irreducible.

I. Introduction

A new horizon for computer industry is in the infancy stage, that is quantum computer. It is different from digital computers, in that the latter, deals with data either as zeros or ones, while in the quantum computation uses quantum bits. Shore's proved that, whenever quantum computer becomes to reality, most cryptography algorithms are cryptanalytic (especially those cryptosystems that depends on factoring, logarithm, and elliptic curve). The only cryptosystem that resist the quantum computer is McEliece Cryptosystem.

McEliece cryptosystem is based on hardness of finding nearest codeword for a linear binary code, which is considering a NP- hard Problem (Non-deterministic Polynomial-time hard),the name stands for McEliece who is suggested it in 1978 [1], to use error correction code in order to send knowledge in a secure method to destination over unsecured channel. This idea is considered out of ordinary due to the main principles of coding theory to ensure that the message received is correct message; while one of the principles of cryptography is protecting the communication over non-secure channels. Error correcting codes harnesses the coding theory in order to detect and fix the errors (as depicted in Fig. 1). The main drawback of error correcting codes is the adding of redundant bits, which makes the message size larger than its original size. The reason behind adding redundant bits is to detect the errors and then fix it.

Fig. 1 Sending message with error correcting

The McEliece cryptosystem suffers from large public key matrix, which leads to be difficult for practical use (with all platforms which have small memories and virtual memories). After that many variants of McEliece cryptosystem were proposed in order to reduce the size of public key [2]-[11]. Unfortunately most proposed system was broken [12]-[15].

Due to the above reasons, McEliece cryptosystem with binary Goppa code have been introduced and studied, which is classified into irreducible and separable binary Goppa code.

A few implementations of original McEliece public key cryptosystem have been proposed, most of them were dealing with a fixed parameters, except an implementation proposed by Repka[16] which deals with unfixed parameters, using C++ language, and he depends on number theory library (NTL), which is use C++ program to factorize, test irreducibility, multiplication, division, and other polynomial operations.

In this paper a new implementation with flexible parameters and dynamic errors, using the two types of binary Goppa code, have been introduced in order to compare between the two types of binary Goppa code. The implementations have been done by graphical user interface (GUI) using Visual Studio C#.

The comparison was done from different perspectives, which are computation time, security, implementation issue, and size of public generator matrix.

II. Binary Goppa Code

The Binary Goppa code is denoted by Γ(g(z); L), where g(z) is a Goppa generator polynomial of degree t over the extension field GF(2^m) and L is the range of code such that L⊆GF(2^m) [17].

$$g(z) = \sum_{i=0}^{t} g_i z^i \quad (1)$$

Thuraya M. Qaradaghi is with the Department of Electrical Engineering, College of Engineering, Salahaddin University-Erbil, Iraq (e-mail: thorayam@gmail.com).

Newroz N. Abdulrazaq is M.Sc student in Department of Computer Science, College of Science, Salahaddin University- Erbil (e-mail: newroz.nooralddin@ gmail.com).

and

$$L = \{\forall\, \alpha_i \in GF(2^m) \backslash g(\alpha_i) = 0\} \quad (2)$$

A. Parameters of Goppa Code

Let $n=2^m$ be the length of codeword c, which is constricted by range L, k is a dimension bounded by $k \geq n - mt$, and minimum distance $d \geq 2t + 1$. Then [n, k, d] denotes to the parameters Goppa code Γ(g(z); L) [1].

A generated Polynomial g(z) is called separable when the polynomial has no roots of multiplicity greater than one (i.e. has no repeated roots). In this case the minimum distance of the Goppa code will be the larger $d \geq 2t + 1$ and can be correct t errors.

Obviously, according (1) and (2) in irreducible Goppa code, none of α's yields the condition $g(\alpha_i) = 0$ i.e. $L=GF\{2^m\}$. While in separable Goppa code, there exist at least ονε 5 σ.τ γ(5)=0.

B. Parity Check Matrix of the Binary Goppa Code

Parity Check matrix H is the most important matrix in encoding and decoding the message. To determine the matrix H:

$$H_{di} = \sum_{\alpha_i \in L} \sum_{k=1}^{t-(d-1)} g_{t-(k-1)} \alpha_i^{t-d+(1-k)} h_i \quad (3)$$

Remark: If c is a codeword, then the parity check matrix H should yield $cH^T=0$.

C. Generator Matrix of the Binary Goppa Code

The generator matrix G of the binary Goppa code used to encode and decode message, while Parity check matrix is important for detecting and correcting errors. The generator matrix G is derived from parity check matrix H, the row space of G is the vectors of nullspace of H modulo 2 such that:

$$GH^T = 0 \quad (4)$$

D. Encoding Message in Binary Goppa Code

Let [n, k, d] be a parameters of Goppa code Γ (L; g(z)), where g(z) is a polynomial of degree *t* over $GF(2^m)$ with range $L \subseteq GF(2^m)$. Then encoding a message by partitioned it into blocks of k bits and multiplying each block with the generator matrix G [17], i.e.:

$$(m_1, m_2, \dots, m_k).G = (c_1, c_2, \dots, c_n) \quad (5)$$

Let $y = (y_1, y_2, \dots, y_n)$ be a received codeword with t errors s.t $d \geq 2t+1$, the following steps must be followed:

1. Calculate parity check matrix $\hat{H}$ for $\Gamma(g^2(z), L)$ using Equation (3).
2. Calculate the Syndrome:

$$S(z) = \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}$$

3. Solving Key Equation using extended Euclidean algorithm:
$\sigma(z)S(z) = \sigma'(z) \pmod{g^2(z)}$ *where* $\sigma(z)$ *is error locator.*
4. Finding the set of error location $B = \{i \;\; s.t\; \sigma(\alpha_i) = 0\}$
5. The error vector $e = (e_1, e_2, \dots, e_n)$ is defined by $e_i = 1$ for $i \in B$ and zeros elsewhere.
6. Determine codeword c= y-e.

Fig. 2 Algorithm of Finding Correcting Errors in Separable Goppa Code

E. Error Correcting of Binary Goppa Code

Finding and fixing errors differs from irreducible than separable Goppa code. Each one have it is own algorithm to fixing errors (as shown in Figs. 2 and 3).

Let $y = (y_1, y_2, \dots, y_n)$ be a received codeword with t errors s.t $d \geq 2t+1$, the following steps must be followed:

1. Calculate parity check matrix H for Γ(g(z), L) using Equation (3).
2. Calculate the Syndrome:

$$S(z) = \sum_{i=1}^{n} \frac{y_i}{z - \alpha_i}$$

3. Find $\sigma(z)$ by follow the following steps:
   - Using extended Euclidean algorithm to determine h(z) such that:
     $S(z)h(z) \equiv 1 \pmod{g(z)} \quad i.e.\, h(z) = S^{-1}(z)$
     If h(z) = z, then the process is finished and $\sigma(z) = z$ elsewhere continue the steps
   - Calculate d(z) such that:
     $d^2(z) \equiv h(z) + z \pmod{g(z)} \quad i.e\; d(z) \equiv \sqrt[2]{h(z) + z} \pmod{g(z)}$
   - Determine a(z) and b(z) using extended Euclidean algorithm:
     $d(z)b(z) \equiv a(z) \pmod{g(z)}$
   - Compute $\sigma(z) = a^2(z) + z\, b^2(z) \pmod{g(z)}$
4. Finding the set of error location $B = \{i \;\; s.t\; \sigma(\alpha_i) = 0\}$
5. The error vector $e = (e_1, e_2, \dots, e_n)$ is defined by $e_i = 1$ for $i \in B$ and zeros elsewhere.
6. Determine codeword c= y-e.

Fig. 3 Algorithm of Finding Correcting Errors in Irreducible Goppa Code

F. Decoding a Message in Binary Goppa Code

When the errors are fixed in codeword, the received message can be decoded [17], using the (5), which can be written as matrix representation:

$$G^T . \begin{pmatrix} m_1 \\ \vdots \\ m_k \end{pmatrix} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$$

For computing the message, Gauss elimination method is applied in order to remove generator matrix G:

$$\left(G^T \middle| \begin{matrix} c_1 \\ \vdots \\ c_n \end{matrix}\right) \sim \cdots \sim \left(\frac{\left(P_I \middle| \begin{matrix} m_1 \\ \vdots \\ m_k \end{matrix}\right)}{P}\right) \quad (6)$$

where I_k is the identity matrix with size k × k and P is a matrix with size (n - k) × (k + 1).

III. McEliece Cryptosystem

The McEliece Cryptosystem is one of the major types of public key cryptosystem. It is classified into three processes (as shown in Fig. 4), namely: Key generation, Encryption process, and Decryption process.

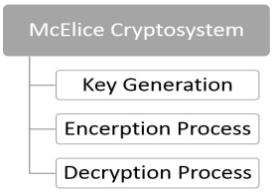

Fig. 4 McEliece Cryptosystem

A. Key Generation of McEliece Cryptosystem

Public key cryptosystem based on two types of keys (public and private), which are linked to get her mathematically. A

public key is published and used to cipher a message, while a private key must be keep it secret and used it to decipher the message [2].To prepare keys depending on Goppa code, the following approaches should be used [1]:

1. The secret key of McEliece PKC depends on three Parameters:

- The first one depends on setup Goppa code, a random polynomial g(z) of degree t over GF(2^m) could be selected. The Goppa code Γ(L; g(z)) has parameters [n; k≥ n -mt; d ≥ 2t + 1]. Based on given parameters calculate the k × n private generator matrix G of the Goppa code as explained in Section II (A-C).
- Pick a random k × k matrix S, s.t S × B= I. The matrix B is derived from Gauss elimination method. This approach are faster than to determining the determinant of a matrix, at the same time reduce the probability of choosing invertible matrix because in our assumption the matrix B is not necessary to be an inverse for picked matrix S.
- Pick an arbitrary n × n permutation matrix P, Where P is a matrix that contains one ones in each row and each column.

2. The Public generator matrix G^* is calculated by $G^* = S \times G \times P$ and should be published with degree of random generator polynomial t.

B. Encryption Process in McEliece Cryptosystem

To encrypt any message, the following steps (as seen in Fig. 5) should be followed:

1. Obviously, in encryption process, there is public generator matrix $G^*_{k\times n}$ and degree of an arbitrary generator polynomial t.
2. Convert each character to decimal number using ASCII code, where each character should have 7 bits length.
3. Collect all binary string together.
4. In case length of message mod k ≠ 0, the message with (k-(length of message mod k)) zero's in the last of the message, should be padding.
5. Each fetching process for k bits from the message should perform steps 6-10.
6. Calculate fetched message × G^*.
7. Create error vector e with size n and include (≤ t) errors (i.e. e has n zero's and convert (≤ t) zero's to one's).

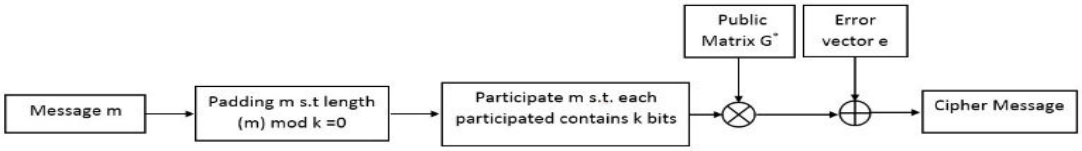

Fig. 5 Encryption Process

C. Decryption Process in McEliece Cryptosystem

To recover plain message from cipher message c, the following steps should be done (as depicted in Fig. 6):

1. The receiver has the following information: Goppa code parameters with secret generator matrix$G_{k\times n}$, Nonsingular Matrix S, and Permutation matrix P.
2. Compute the invertible of matrix S, and the inverse of Permutation matrix.
3. Partition the cipher message into parts, where each part contains k bits.
4. For each entity should perform steps 5-9.
5. Compute $mSG' = c \times P^{-1} = mSGP + P' = mSG + e'$.
6. Use efficient decoding algorithm (if the Goppa code is separable (as shown in Fig. 2) or irreducible (as shown in Fig. 3) to find error location e'.
7. Calculate $mSG = mSG' + e'$.
8. Removing secret generator matrix G using Gaussian elimination method to get (mS). (as explained in Section II (F).
9. Compute$m = mS \times S^{-1}$.
10. Collect all computing message together.
11. Make sure length (message) mod 7 =0; in case of inequality we could remove (length (message) mod 7) zero's from the last of the message.
12. Fetch every time 7 bits from the message and convert it to decimal number.
13. Convert each decimal number to character using ASCII code table, and then collect all characters together to obtain the plain message.

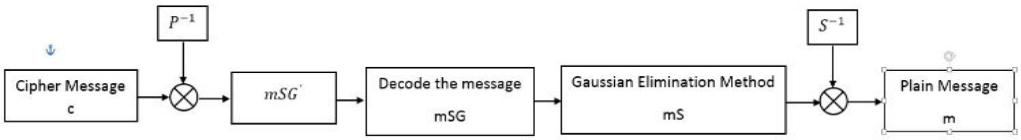

Fig. 6 Decryption Process

IV. A Prototype for Designed System Using Visual Studio C#

The designed platforms have been done by three stages using Visual Studio c#, the first one is to generate secret and public key for the required cryptosystem, and the second stage is for encryption process, while the last one used to decrypt the message. It has the following specifications:

1. The system randomly generate a Goppa code Γ(L; g(z)) and then tested it, if the code match the condition of Goppa code (Irreducible or separable) then it starts the process, whereas the system will inform the user why the condition dose not yield then start to generate a new code.
2. The designed system deals with flexible parameters and dynamic error vectors. The dynamic error (dynamic errors means that the sender have the right to choose number of errors less than from published one without notifying the receiver and for the next block of message the error locations are changed randomly with a new number of errors). This process increases the time attacks (which are based on finding minimum codewords) against McEliece cryptosystem.
3. The designed system, record every details and operations required by McEliece cryptosystem (for key generation, encryption, and decryption process) in text files. This process helps the researchers to do a well studying for the designed cryptosystem and it is useful for teaching propose.

A. Key Generation Stage

In this stage many forms has designed in order to determine each needed operation separately. The designed system begin from factoring polynomial and testing irreducibility and ended with generating public generator matrix as explained in Section III (A). Figs. 7-9 declare how the key generation stages are designed. These figures are part of several forms.

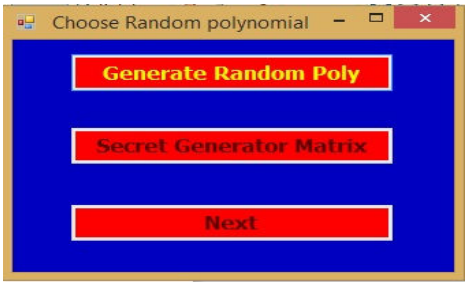

Fig. 7 Generating Secret Matrix

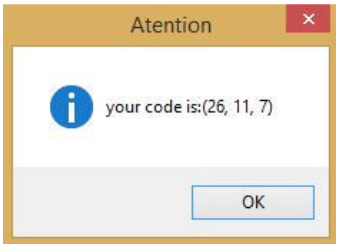

Fig. 8 Parameters of Code

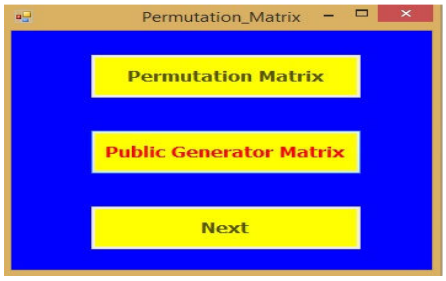

Fig. 9 Generating Public Matrix

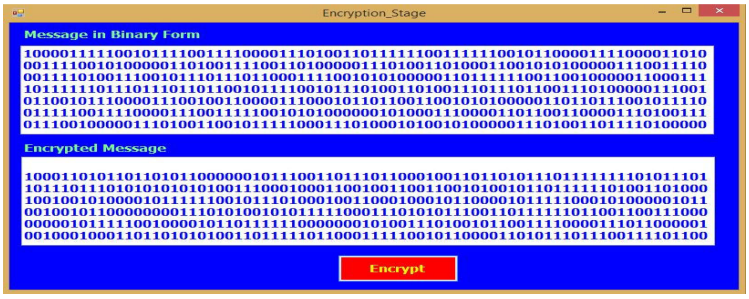

Fig. 10 Encryption Stage

B. Encryption Stage

This stage consists of three forms, the two forms used to enter the message in two ways. The First way is about import file from specified folder, while the second form is entering the message within the textbox. During the two forms the message has partitioned and converted to a numbers using ASCII code. Whereas the third form begin to encrypt the message as shown in Fig. 10.

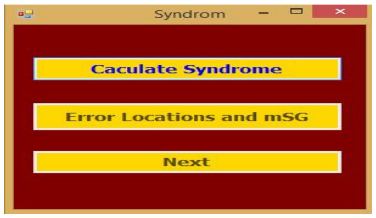

Fig. 11 Syndrome and Error Locations Form

C. Decryption Stage

In this stage several forms have been designed in order to determine each needed operations separately. The designed system begin from determine parity check matrix in case of separable Goppa code, decoding algorithm, etc. as explained in Section III (C). Figs. 11 and 12 show how the decryption stages are designed. These figures are part of several forms.

Fig. 12 Decryption Stage

V. Comparison between Separable and Irreducible Goppa Code in McEliece Cryptosystem

A. Collecting Data

In order to compare between separable and irreducible Goppa code, two measurement types have been recorded. The first one records the average computation time for Parity check matrix (in case of separable type, time computation in encryption and decryption process have been counted), error locator polynomial (sigma), and encryption stage for (50 Byte) message.

TABLE I
Computation Time of Parity Check Matrix for Separable and Irreducible Goppa Code\CPU Ticks

Degree of g(z) t	Extension number (m)	Av. Separable\ CPU Ticks	Av. Irreducible\ CPU Ticks
2	4	1.26E+04	4.64E+03
3	4	1.75E+04	6.74E+03
4	5	6.47E+04	2.30E+04
5	5	6.55E+04	2.81E+04
6	6	2.86E+05	8.94E+04
7	6	3.64E+05	9.65E+04
8	7	2.55E+06	6.70E+05
9	7	2.35E+06	5.63E+05
10	8	1.19E+07	3.22E+06
11	8	1.46E+07	3.60E+06

TABLE II
Computation Time of Error Locator for Separable and Irreducible Goppa Code\CPU Ticks

Degree of g(z) t	Extension number (m)	Av. Separable\ CPU Ticks	Av. Irreducible\ CPU Ticks
2	4	7.01E+02	1.43E+03
3	4	1.03E+03	2.12E+03
4	5	2.23E+03	4.58E+03
5	5	3.01E+03	5.31E+03
6	6	6.85E+03	1.35E+04
7	6	7.57E+03	1.83E+04
8	7	3.45E+04	1.06E+05
9	7	3.11E+04	1.04E+05
10	8	1.25E+05	8.37E+05
11	8	1.60E+05	8.37E+05

TABLE III
Computation Time to Encrypt (50 Byte) Message for Separable and Irreducible Goppa Code\CPU Ticks

Degree of g(z) t	Extension number (m)	Av. Separable\ CPU Ticks	Av. Irreducible\ CPU Ticks
2	4	1.42E+05	8.28E+04
3	4	4.74E+05	3.23E+05
4	5	6.10E+02	6.47E+04
5	5	3.20E+05	1.61E+05
6	6	2.13E+04	2.22E+04
7	6	4.88E+04	3.65E+04
8	7	1.58E+04	1.98E+04
9	7	2.26E+04	2.01E+04
10	8	1.42E+04	1.77E+04
11	8	1.64E+04	1.98E+04

The designed system is implemented for random parameters m=4, 5, ...,8 and for each extension number (m), two degree of random generator polynomial is implemented, which is starts from 2 to 11 respectively (as shown in Tables I-III).

TABLE IV
Computation Time to Encrypt Message for Separable and Irreducible Goppa Code\CPU Ticks

Message Size\ Byte	Av. Separable with One Root	Av. Separable\ CPU Ticks with t root	Av. Irreducible\ CPU Ticks
118	8.36E+04	5.14E+04	8.54E+04
223	1.78E+05	1.29E+05	1.74E+05
348	3.18E+05	2.48E+05	2.95E+05
413	6.12E+05	3.52E+05	4.50E+05
558	8.31E+05	6.73E+05	8.70E+05
657	1.15E+06	9.50E+05	1.21E+06
751	1.76E+06	1.31E+06	1.53E+06
836	2.01E+06	1.75E+06	2.03E+06
964	2.58E+06	2.26E+06	2.66E+06
1136	4.17E+06	3.74E+06	4.17E+06

While second measurement records the computation time for encryption process with different random message size,

which is start from 100 Byte to 1kB. Due to yield notable chart, smaller Goppa code (m=8, t=10) have been implemented (as shown in Table IV).

A. Analyzing Time Computation

As shown in Fig. 13, which is derived from Table I, the computation time for calculating parity check matrix is higher than the time needed to calculate it in irreducible McEliece cryptosystem. It is expected results due to determining additional parity check matrix in decryption process for $g^2(z)$ in separable type. Actually this is one of the reasons behind preference irreducible type over separable type.

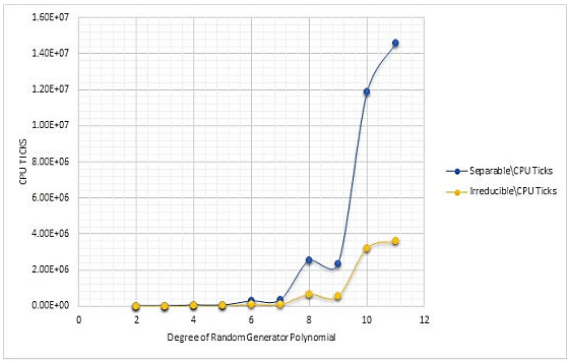

Fig. 13 Average Computation Time of Parity Check Matrix in Respect to Degree of g(z) for Separable and Irreducible Types

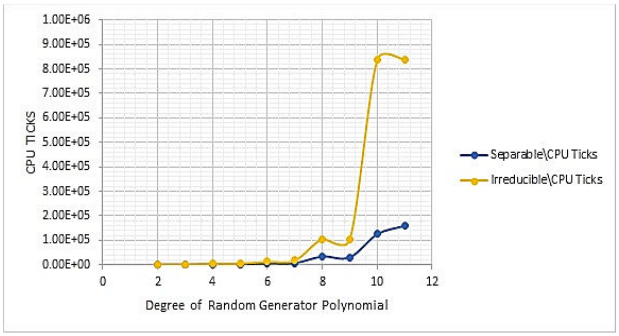

Fig. 14 Average Computation Error Locator Polynomial (Sigma) in Respect to Degree of g(z) for Separable and Irreducible Types

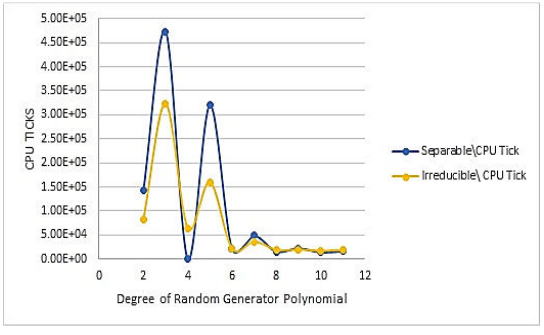

Fig. 15 Average Computation Time of Encryption Process for (50 Byte) Random Message in Respect to Degree of g(z) for Separable and Irreducible Types

Fig. 14 shows that the time needed to execute error locator in decryption process for separable type, is better than the computing time in irreducible type. For encryption stage, to get better result for comparison, two types of testing have been chosen, in the first one the random message is constant while the parameter of Goppa code are changed. But in the second test, the parameters of Goppa code are constant (m=8 and t=10), while the random message are changed. The result shows that the time needed to execute it is closed especially for big (m)'s (choosing big (m)'s are better for security propose) (as shown in Figs. 15 and 16)).

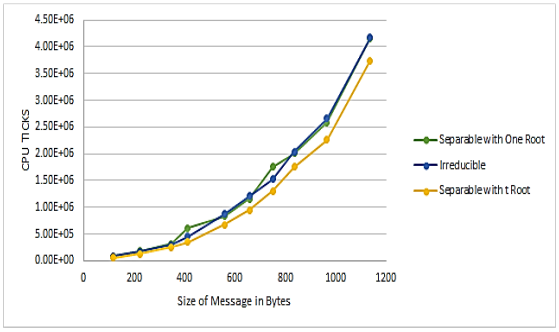

Fig. 16 Average Computation Time of Encryption Process in Respect to Random Message for Separable and Irreducible Types

VI. Several Perspectives on Comparison between Separable and Irreducible Goppa Code in McEliece Cryptosystem

In general, the separable and irreducible McEliece cryptosystem can be compared in four perspectives, as below:

1. Time Computation

It is clear, the consuming time are closed to be balance, which is disprove that vision about preference irreducible over separable type.

2. Security

Till now, there is no study mentioned that there is an active attack on separable type except that attacks on cyclic and dyadic Goppa code which is depend on separable type (the attack depends on secret matrix which is generated by cyclic the first row of matrix while in the designed system, the secret matrix are generated by the null space).

3. Implementation Issues

Due to calculating a parity check matrix for $g^2(z)$, it may cause a problem with enlarge t in implementation time (i.e. maximize the size of matrix from (t× no. of columns) to (2t × no. of columns), which may cause a problem with big t.

4. Memory

The size of generator public matrix in original McEliece cryptosystem is large, which is considering an effective drawback. Using separable Goppa code, reduces the size of public key, for example if m=8 and t=10 have been selected, the size of public key in irreducible type [256, 176, 21] will be

256 × 176 = 45056 bits, while in separable type [246, 166, 21] will be 246 × 166 = 40836 bits

VII. Conclusions

A graphical user interface of McEliece public key cryptosystem, have been designed using the two types of Goppa code (irreducible and separable) with unfixed parameters and dynamic errors. The designed system increases the attacking time against attacks based on finding minimum codeword. Also, a comparison between separable and irreducible have been done, and founded in general implementation the two types are closed to be balanced. Separable type may cause a problem in implementation time for those programming languages that deals with smaller size of integer data types (which is convert to exponential format), whenever degree of generator polynomial are big. On the other hand, separable type needs less memory than irreducible Goppa code to store public generator matrix.

References

[1] R. McEliece. A Public-Key Cryptosystem Based on Algebraic Coding Theory.Technical report, NASA, 1978.
[2] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani. A Variant of the McEliece Cryptosystem with Increased Public Key Security. Workshop on Coding and Cryptography WCC 2011, pages 173-182, Paris, France, Apr. 2011.
[3] M. Baldi, F. Chiaraluce, R. Garello, and F. Mininni. Quasi-Cyclic Low-Density Parity-Check Codes in the McEliece Cryptosystem. In ICC, pages 951-956. IEEE, 2007.
[4] T. P. Berger, P.-L. Cayrel, P. Gaborit, and A. Otmani. Reducing Key Length of the McEliece Cryptosystem. In B. Preneel, editor, AFRICACRYPT, volume 5580 of Lecture Notes in Computer Science, pages 77-97. Springer, 2009.
[5] Edoardo Persichetti, Compact McEliece Keys Based on Quasi-Dyadic Srivastava Codes. IACR Cryptology ePrint Archive, 2011 (preprint).
[6] E. M. Gabidulin, A. V. Ourivski, B. Honary, and B. Ammar. Reducible rank codesand their applications to cryptography. IEEE Transactions on Information Theory, 49(12):3289-3293, 2003.
[7] E. M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a Non-Commutative Ring and their Applications in Cryptology. In D. W. Davies, editor EUROCRYPT, volume 547 of Lecture Notes in Computer Science, pages 482-489. Springer, 1991.
[8] R. Misoczki and P. S. L. M. Barreto. Compact McEliece Keys from Goppa Codes. In M. J. Jacobson Jr., V. Rijmen, and R. Safavi-Naini, Selected Areas in Cryptography, volume 5867 of Lecture Notes in Computer Science, pages 376-392. Springer, 2009.
[9] C. Monico, J. Rosenthal, and A. Shokrollahi. Using low density parity check codesin the McEliece cryptosystem. In IEEE International Symposium on Information Theory, ISIT 2000, page 215. IEEE, 2000.
[10] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. Problemsof Control and Information Theory, 15(2):159-166, 1986.
[11] V. M. Sidelnikov. A public-key cryptosystem based on binary Reed-Muller codes.Discrete Mathematics and Applications, 4(3):191-208, 1994.
[12] J. -C. Faug*e′*re, A. Otmani, L. Perret, and J. -P. Tillich. Algebraic Cryptanalysis of McEliece Variants with Compact Keys. In H. Gilbert, editor, EUROCRYPT, volume 6110 of Lecture Notes in Computer Science, pages 279-298. Springer, 2010.
[13] L. Minder and A. Shokrollahi. Cryptanalysis of the Sidelnikov Cryptosystem. InM. Naor, EUROCRYPT, volume 4515 of Lecture Notes in Computer Science, pages 347-360. Springer, 2007.
[14] R. Overbeck. A New Structural Attack for GPT and Variants. In E. Dawson and S. Vaudenay, editors, Mycrypt, volume 3715 of Lecture Notes in Computer Science, pages 50-63. Springer, 2005.
[15] V. M. Sidelnikov and S. O. Shestakov. On insecurity of cryptosystems based on generalized Reed-Solomon codes. Discrete Mathematics and Applications, 2(4):439-444, 1992.
[16] Repka Marek, McEliece PKC Calculator, Journal of Electrical Engineering, volume 65, Issue 6, 342-984,Nov,2014.
[17] Wade Trappe, Lawrence Washington, Introduction to Cryptography with Coding Theory, 2nd ed., USA Pearson, 2006.