Journal of
Engineering and
Applied Sciences

SINCE 2000
MEDWELL PUBLICATIONS

McEliece in RADG using Diffie-Hellman Key Exchange Security System

Salah Albermany and Zahraa Naseer
Faculty of Mathematics and Computer Science, University of Kufa, Kufa, Iraq

Key words: Public key, RADG, Diffie-Hellman key exchange, security

Corresponding Author:
Salah Albermany
Faculty of Mathematics and Computer Science, University of Kufa, Kufa, Iraq

Page No.: 3138-3145
Volume: 15, Issue 17, 2020
ISSN: 1816-949x
Journal of Engineering and Applied Sciences
Copy Right: Medwell Publications

Abstract: A high-performance ciphering algorithm is presented. The proposed method combines the ciphering technique (Reaction Automata Direct Graph (RADG)) with McEliece cryptosystem to obtain higher level of security. This proposed algorithm has dynamic transitions too generated by Deffie-Hellman key exchange. McEliece algorithm is an asymmetric encryption algorithm, adding a higher level of security to the proposed approach called ME-RADG-Diffie-Hellman.

INTRODUCTION

The need for security aspects has increased in the current era because of overlapping information and access to information via. the internet, this information need a modern and developed means to protect to ensure that no manipulation of information and change as well as when sending messages between individuals or official bodies these messages needs to the means to protect their own. To prevent penetration or get this information from these methods of encryption, the encrypt of information is ways proved effective in making or delay the discovery of information (to delay the discovery of information when the information becomes relevant non-values or unimportant)[1]. The encryption process in the system proposed by the algorithm of McEliece cryptosystem, it is an asymmetric encryption algorithm using the public key where everyone is dealing with the public and private key When you send a message the sender must have the private key and the recipient must have a private key and a public key where the sender sends the encrypted message which encrypted with recipient public key and when the recipient gets the message he will decrypt this message with his private key. We conclude from these words that the sender has a public key and private revenue as well as his public key to the recipient[2]. This study using presents a novel keyless security scheme which is based on Reaction Automata Direct Graph (RADG)[3], a new trend in security that has recently merged and proved to be efficient. However, one of the weaknesses of this technique is the fixed graph design but proposed algorithm depends on McEliec key with dynamic transitions generated by using the Deffie-Hellman key exchange. The problem discussed in this study is changing the graph from fixed to dynamic using the Deffie-Hellman key exchange, decreasing the decryption time yet adding more security level.

RADG: RADG (Reaction Automata Direct Graph) is algorithm which does not need a key exchange between the two parties because of the combination of automatic graph and reaction states. RADG design consists of a sextuple represents as (Q; R; Σ; Ψ; J; T) where: 'Q' stands for a set of standard states, 'R^' stands for a set of reaction states, 'Σ' stands for a set of input data, 'Ψ' stands for a set of output transitions, 'J' stands for a set (which is subset of Qcalled jump states) and 'T' Represents transition Function. Each state has λ values. RADG encryption is based on the relationship between states, from another side, the design of RADG is based on m, n, k and λ where n (the size of the Q-set) = |Q|, m (the size of the R-set)= |R| and λ represents the number of values in each state except jump set states. The encryption starts with Q-set of states with each value of the state depending on the message and when the transitions get to a jump state they will move randomly and taking the values from the R-set and going back to the Q-set using the corresponding transition. The example below helps understanding RADG design must $n \geq 2$ and $k \leq \lfloor n/2 \rfloor$[3].

Example: If $n = 4$, $k = 1$, $m = 1$ and suppose $\lambda = 2$ number of value in each state then the transition among states as shown in As shown in Fig. 1, if $\lambda = 2$, then $\Sigma = \{0, 1\}$, number of states in R (reaction states) is only one state according to m = 1, number of jump states is 1 according k = 1 and number of states in Q (standard states) is 4 according to n = 4 and $\Psi = \{15\text{-}24\}$. Suppose the original message to be encrypted using RADG is 1110. Transition function as T (address of state bit of message). Thus, we have T(0.1) = (3.18), T(3.1) = (2.20), T(2.1) = (1.22) and T(1.0) = (3.15). The corresponding outputs are 18, 20, 22, 15, respectively.

McEliece cryptosystem: The McEliece cryptosystem is a type of public key cryptosystem that uses a linear error-correcting code in order to create a public key and a private key. The original error correcting code used in this cryptosystem is the binary Goppa code. A public key, as one would assume is public anyone and everyone can find it. In order to do this, the private key is only held by the receiver of the message. Traditionally, we use the example of two friends, Alice and Bob, to explain cryptography. Suppose Alice wants to send a private message to Bob. Bob must first publish his public key, which is based on his private key. Then, Alice takes Bob's public key and encrypts her message with it. The message then becomes a codeword. She sends her encrypted message to Bob. Bob then uses his private key to decrypt the codeword and read the message[4].

In order to construct the public and private keys, Bob must first choose an arbitrary Goppa polynomial g(z) with a degree t over $GF(m^2)$. The Goppa code is defined by this polynomial and by L has parameters $[n, \geq n\text{-}mt, \geq 2t+1]$. Using this, Bob would then compute the $k \times n$ generator matrix G of the Goppa code. Then, Bob randomly chooses a $k \times k$ invertible matrix S and a $n \times n$ permutation matrix P which means that P has exactly 1 in every row and column with all other entries being zero. Then he computes G' = SGP.G' is his encoding matrix. This results in his public key consisting of G' and t only[5].

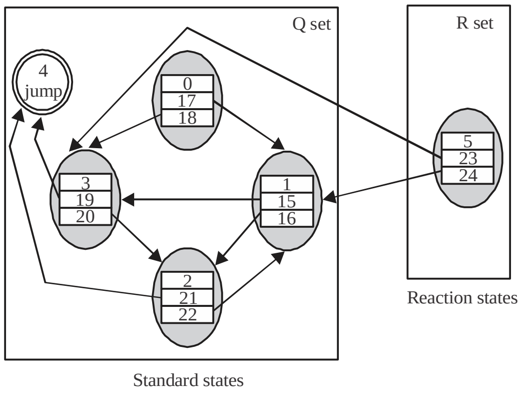

Fig. 1: Example of flowing ciphering data in RADG design

The private key consists of the polynomial g(z), the original matrix G, along with matrices S and P such that:

$$G' = SGP \quad (1)$$

Once Bob publishes his public key, Alice generates a random binary vector e of length k that has a weight $wt(e) \leq t$. Then, Alice can encode her message m = (m_1, m_2, …, m_k) by computing:

$$y = mG'+e \quad (2)$$

Then, Alice sends her ciphertext y. Bob receives Alice's codeword and uses his permutation matrix P to compute $y' = yP(-1) = mG'P^{(-1)}+eP^{(-1)} = mSG\ PP^{(-1)}+e' = (mS)G+e'$.

Bob can then decode y' into the message m' = mS by finding e' which is done by Bob applying Patterson's algorithm[6]. Once this is done, Bob can calculate y-e' = mSG and since, Bob knows what S is he can calculate $S^{(-1)}$ and then recover the original message:

$$m = m'S^{(-1)} \quad (3)$$

Basically, the parameters of the McEliece crypto system are the parameters of the Goppa code used. After choosing the underlying Galois field $GF(m^2)$ and the error correcting capability of the code t, all other parameters depend on these two values. The original parameters suggested by McEliece are m = 10, t = 50 but in citehac, the authors noted that t = 38 is a better choice with regard to the computational complexity of the algorithm without reducing the security level. These parameters nowdays only give 2^{60}-bit security compared to a symmetric ciher. To achieve an appropriate level of 2^{80}-bit security at least parameters $n = 2^{11} = 2048$, t = 38 and $k = n\text{-}m\cdot t = 1751$ must be chosen.

Diffie-Hellman key exchange: The purpose of DH algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm is itself limited to the exchange of secret values.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. Primitive root of a prime number P is one whose powers modulo P generate all the integers from 1 to -1. That is, if is a primitive root of the prime number P, then the numbers a mod p, a^2 mod p, …, $a^{(p-1)}$ mod p are distinct and consist of the integers from 1 through p-1 in some permutation. For any integer b and a primitive root a of prime number, we can find a unique exponent i such that:

$$b \equiv a^i (\text{mod } p) \tag{4}$$

where, $0 \leq i \leq (p-1)$. The exponent i is referred to as the discrete logarithm of b for the base, mod p. Discrete logarithms are logarithms defined with regard to multiplicative cyclic groups. If G is a multiplicative cyclic group and g is a generator of G, then from the definition of cyclic groups, we know every element h in G can be written as gx for some x. The discrete logarithm to the base g of h in the group G is defined to be x. For example, if the group is Z^*_5 and the generator is 2, then the discrete logarithm of 1 is 4 because 24 = 1 mod 5[2].

The proposed ME-RADG-Diffie-Hellman: ME-RADG-Diffie-Hellman is (McEliece in RADG using Diffie-Hellman key exchange security system). This proposed method is an asymmetric encryption algorithm, bused on RADG method, McEliece and Diffie-Hellman key exchange. The Diffie-Hellman key exchange is using to generate a dynamic transition design.

The ME-RADG-Diffie-Hellman design is based on RADG mathematical model and the characteristics of the RADG algorithm. ME-RADG-Diffie-Hellman contain 6 tuples as original RADG which are {R, Q, Σ, Ψ, J, T} as it has been discussed in (section II) as well as the Deffie-Hellman key exchange in (section IV).

Diffie-Hellman key exchange is used as the first step in the transition function which allows both sender and receiver to generate the next states.

Encryption: The first step in encryption process, encrypt the message by using the ME-RADG-Diffie-Hellman algorithm is convert the message in to binary value. Let, the binary Message M:

$$M = m_1, m_2, m_3, m_4, m_5, ..., m_j, ..., m_{|M|}$$

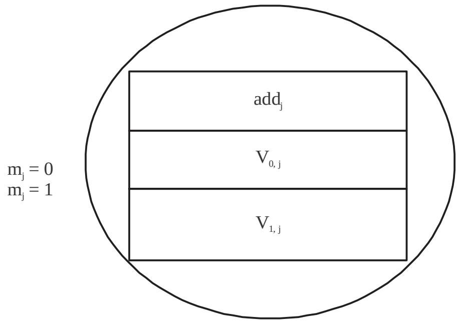

Fig. 2: Method of choosing the index of value from state

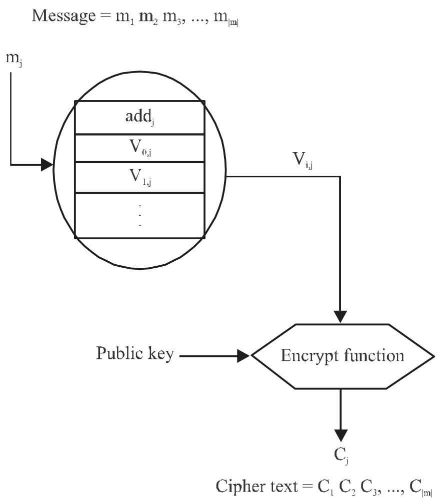
Message = m_1 m_2 m_3, ..., m_|m|
m_j
add_j
V_0,j
V_1,j
V_i,j
Public key
Encrypt function
C_j
Cipher text = C_1 C_2 C_3, ..., C_|m|

Fig. 3: General design of ME-RADG-Diffie-Hellman encryption

where, M_j = 0, 1(j = 1, 2, …, |M|). From the transition function get the address of first state (add_1). Every state has set of value $V_{(i, j)}$ (i = 0, 1, …, λ-1).

Let λ = 2, value = {V_0, V_1} OR value (add_1) = {$V_{0, 1,}$ $V_{1, 1}$}. Where add_1 corresponding m_1, add_2 correspond in m_2, …, add_j corresponding m_j. In general value (add_j) = {$V_{0, j}$, $V_{1, j}$, …, $V_{\lambda-1, j}$}. Then encryption of m_1 is $V_{i,1}$ (Fig. 2).

The output value gets encrypted using McElice using public key matrix G' (the dimensions of G' matrix = $k_{mc} \times nm_c$):

$$C_j = \text{McEliece}(V_{i, j})$$

Encrypted ($V_{i, 1}$) by McEliece algorithm by using the public key (G'): $[P_{1_{kmc\ bit}} \times \tilde{G}_{k_{mc \times nmc}} = y_{n_{mc\ bit}}]$. Then by adding a random error vector (e) of length n_{mc} bit and weight(t) then C_1 = y+e repeating above process for to C_1, C_2, …, to $C_{|m|}$ where, $C_{|m|}$ = McEliece ($V_{(i, |m|)}$). Then, the Cipher text is C_1, C_1, ..., $C_{|m|}$. Clarifying the above steps in Fig. 3.

Transition function: This function generates the next states which add an advantage to the previous method making it dynamic transitions without the need of adding the address to the cipher. The first step selects a large prime number (q) and select number ($\propto$), $\propto$ must be a primitive root of q and ($\propto$<q) [agreement between Alice and Bob on q and $\propto$]. The output must be within the J-set or Q-set. This is done by using this condition ($|Q| \leq q-1$). The sender selects the privet key X_A such that (X_A<q) and calculates the public key Y_A using $Y_A = \propto X_A$ mod q.

The receiver selects the privet key X_B such that (X_B<q) and calculates the public key Y_B using $Y_B = \propto X_B$ mod q. The sender sends Y_A to the receiver and the receiver sends Y_B to the sender. The sender calculates the secret key transition by equation:

$$\text{Secret key transition of sender} = {Y_B}^{X_A} \bmod q$$

The receiver calculates the secret key transition by equation:

$$\text{Secret key transition of receiver} = {Y_A}^{X_B} \bmod q$$

Must the secret key transition of sender = secret key transition of receiver. When encryption process using the secret key transition of sender to calculates the address of first state (add_1) by using the equation:

$$add_1 = \propto^{\text{Secret key transition of sender}} \bmod q$$

If the add_1 from the Q set calculate the address of second state (add_2) by using the equation:

$$add_2 = \propto^{\text{Secret key transition of sender+1}} \bmod q$$

Else the add_1 from the J set then select random state from R set. This state becomes add_1 and calculates the address of second state (add_2) by using the equation:

$$add_2 = \propto^{\text{Secret key transition of sender+1}} \bmod q$$

In general $add_j = \propto^{\text{Secret key transition of sender+j-1}}$ mod q (j = 1, 2, …, q-2). When decryption process using the secret key transition of receiver to calculates the address of first state (add_1) by using the equation:

$$add_1 = \propto^{\text{Secret key transition of receiver}} \bmod q$$

If the add_1 from the Q set calculate the address of second state (add_2) by using the equation:

$$add_2 = \propto^{\text{Secretkey transition of receiver+1}} \bmod q$$

Else the add_1 from the J set then select random state from R set. This state becomes add_1 and calculates the address of second state (add_2) by using the equation:

$$add_2 = \propto^{\text{Secret key transition of receiver+1)}} \bmod q$$

In general, $add_j = \propto^{\text{Secret key transition of receiver+j-1}}$ mod q (j = 1, 2, …, q-2). Repeat these operations to end the message, {add_j}, j = 1, 2, …, |M|.

Decryption: The first step decrypts process, decrypt the message by using (The ME-RADG-Diffie-Hellman) algorithm is read the cipher text.

Let the cipher text C = $c_1, c_2, c_3, c_4, c_5, ..., c_j, ..., c_{|M|}$. The second step decrypts the first value in cipher text (c_j) by McEliece algorithm using privet key matrixes (P, G, S, t), the dimensions of (P matrix = $n_{mc} \times n_{mc}$), (G matrix = $k_{mc} \times n_{mc}$) and (S matrix = $k_{mc} \times k_{mc}$) (the input is n_{mc} bit and output is k_{mc} bit:

$$D_j = \text{McEliece}(c_j)$$

Let decrypted (c_1) by McEliece algorithm using the privet key (P, G, S, t), the result is D_1 (k_{mc} bit). Third step from the transition function gets the address of first state is (add_1). Every state has a set of values $V_{i,j}$ (i = 0, 1, …, λ-1).

Let λ = 2, value = {V_0, V_1} OR value (add_1) = {$V_{0,1}$, $V_{1,1}$} where add_1 corresponding D_1, add_2 corresponding D_2, ..., add_j corresponding D_j

If $D_1 = V_{0,1}$ then $m_1 = 0$, else $D_1 = V_{1,1}$, then $m_1 = 1$. Repeating this process to the end of cipher text, then the message M = $m_1, m_2, ..., m_{|M|}$.

Algorithms: This section provides algorithms of ME-RADG-Diffie-Hellman encryption and decryption as well as algorithm of the transition function where substitution is calculated in two directions (sender and receiver) such that the operations of the transition function efficiently in the inverse direction.

Encryption: Encryption process using ME-RADG-Diffie-Hellman algorithm encrypts input message M = {$m_0, m_1, ..., m_{|M|}$} and uses the McEliece algorithm by using the public key (Algorithm 1).

Algorithm 1; ME-RADG-Diffie-Hellman encryption:

```
Input: Message (M)
Output: cipher text (C) ={$c_1, c_2, …, c_{|c|}$}
1: read binary message M = {$m_1, m_2, ..., m_{|m|}$}
2: While (no<message length)
3: $state_{no} \leftarrow$ Transition function
4: if $state_{no}$ from Q set
5: value $\leftarrow$ Q[$state_{no}$]: get value [message]
6: convert the value from decimal to binary vector
```

```
7: cipher[no]←McEliece E (value)
8: else $state_{no}$ from J set
9: State no random (0, R length)
10: value←R[$state_{no}$]: get value [message]
11: go to step 6 and 7
12: End if
13: End while
14: return cipher text←cipher [no]
End algorithm
```

Transition function: This function works when encryption and decryption (by sender and receiver) (Algorithm 2).

Algorithm 2; ME-RADG-Diffie-Hellman transition function:

```
Input: Q, J and R-set
Output: addresses of state.\\{$state_{no}$} no = 1, 2, …, |M|
1: select q and ∝, must ∝ is a primitive root of q and (∝<q) [agreement between Alice and Bob on q and ∝]
2: Alice selects the privet key $X_A$ such that ($X_A$<q)
3: Alice calculates the public key $Y_A$ using $Y_A = \propto^{X_A} \bmod q$
4: Bob select the privet key $X_B$ such that ($X_B$<q)
5: Bob calculate the public key $Y_B$ using $Y_B = \propto^{X_B} \bmod q$
6: (Alice send the public key ($Y_A$) to Bob)
7: (Bob send the public key ($Y_A$) to Alice)
8: Alice Secret key transition of Alice←$Y_B^{X_A}$ mod q
9: Bob secret key transition of Bob←$Y_A^{X_B}$ mod q
10: While (no<message lenght)
11: when Alice needed Generating a series of states for j←1 to q-2
    $state_{no}$←$\propto^{\text{Secret key Transition of Alice+j-1}}$ mod q
    end for
12: when Bob needed Generating a series of states
    for j←1 to q-2
    $state_{no}\propto^{\text{Secret key transition of Bob+j-1}}$ mod q
    end for
14: end While
15: return $state_{no}$\\state number
End algorithm
```

Decryption: The decryption process of ME-RADG-Diffie-Hellman algorithm is based on the same structure of encryption process but depends on the privet key in inverse process to decrypt cipher and using McEliece algorithm (Algorithm 3).

Algorithm 3; ME-RADG-Diffie-Hellman decryption:

```
Input: ciphertext C = {$c_0$, $c_1$,..., c|c|}
Output: Message (M)
1: read the cipher text C = {$c_0$, $c_1$, ..., c|c|}
2: While (no<message length)\\no is index of cipher
4: convert the cipher of value ($c_{no}$) from decimal to binary vector
5: value[no]←McElieceD($c_{no}$)\\decrypt the cipher of value ($c_{no}$) by using McEliece
6: $state_{no}$←Transition function
7: if $state_{no}$ from Q set
8: decipher[no] = Q[$state_{no}$]: get value [message]
9: else
10: decipher[no] = R[$state_{no}$]: get value [message]
11: End if
12: End while
13: return message M←decipher[no]
End algorithm
```

Table 1: Values of R and Q set (in ME-RADG-Diffie-Hellman example)

State	Cipher if 0	Cipher if 1	Status
1	2	3	Q
2	6	7	Q
3	8	4	Q
4	10	5	Q
5	11	1	Q
6	Jump	Jump	J
7	0	3	Q
8	9	13	Q
9	6	14	Q
10	Jump	Jump	J
11	7	12	Q
12	5	15	Q
13	2	8	R
14	6	3	R
15	10	5	R
16	7	9	R
17	1	4	R
18	0	11	R
19	13	15	R
20	14	12	R

Example: Assuming the two users agreed on the linear code C is (7, 4) and generator matrix:

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}_{4\times 7}, \ t = 1, \ x = 1$$

and the message M is (1 1 0 1 0 0 1) encrypts the message M by using ME-RADG-Diffie-Hellman cryptosystem. Given the cipher in Table 1.

Generate of the keys: The n_{mc}=7, k_{mc} = 4 (from the linear code C). We need to choose our random matrices S (binary non-singular matrix) must be ($k_{mc}\times k_{mc}$) matrix and (binary permutation matrix) must be ($n_{mc}\times n_{mc}$) matrix choosing our random matrices as follows:

$$S = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}_{4\times 4}$$

$$P = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}_{7\times 7}$$

Compute the public key G' = SGP

$$G^{'} = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}_{4\times 7}$$

let q = 13, q is a prime number (agreement between two users the sender and receiver):

$$q\text{-}1 = 12 = |Q|$$

let $\propto = 2$ [is primitive root of q and $\propto<q$]. Let, the sender called Bob and the receiver called Alice. Let Alice select the privet key $X_A = 3$, ($X_A<q$). Alice calculates the public key Y_A using $Y_A = \propto^{X}{}_{A}$ mod q = 2^3 mod 13 = 8. Let Bob select the privet key $X_B = 7$, ($X_B<q$).

Bob calculate the public key Y_B using $Y_B = \propto^{X}{}_{B}$ mod q = 2^7 mod 13 = 11 the public key of receiver (Alice) is (G', t, Y_A). (Alice send the public key to Bob) the private key of receiver is (S, G, P, X_A). The public key of sender (Bob) is (Y_B). (Bob send the public key to Alice) the private key of sender is (X_B).

Encryption: The encryption of the message depends on 2 functions: encrypt function to encrypt the values and transition function to transition between the states.

Transition function: |Q| = 12 (Number of Q-set and J-set (|J|⊆|Q|)) |J| = 2 (Number of J-set), |R| = 8 (Number of R-set) Must $|Q|\leq q\text{-}1$ (because the output must be within the J-set or Q-set) secret key transition of Bob (sender) = Y_A^{XB} mod q = 8^7 mod 13 = 5 address of first state = $\propto^{\text{Secret key transition of Bob}}$ mod q = 2^5 mod 13 = 6.

Since, the address of first the state is 6 and the state 6 is a jump state, then select a random state from R set, for example, select state 16, then $add_1 = 16$ address of second state = $\propto^{\text{Secret key transition of Bob+1}}$ mod q = 2^6 mod 13 = 12. Since, the address of second the state is 12 and the state 12 is from Q-set, then $add_2 = 12$ address of third state = $\propto^{\text{Secret key transition of Bob+2}}$ mod q = 2^7 mod 13 = 11.

Since, the address of third the state is 11 and the state 11 is from Q-set, then $add_3 = 11$. Repeating this process to the end of message, the result is {add_1, add_2, add_3, add_4, add_5, add_6, add_7} = {16, 12, 11, 9, 5, 20, 7}.

Encrypt the values: Since, |Q|+|J| = 10+2 = 12 and q-1 = 13-1 = 12, then |Q|+|J| = q-1. Since, x = 1, then $\lambda = 2^x = 2$ (Each state has two values).

Since, $k_{mc} = 4$, then $2^{kmc} = 2^4 = 16$ all the values$\leq 2^{kmc}$ (Because it needs to be encrypted with a public key matrix G', the size of G' is $k_{mc}\times n_{mc}$.

Step 1: Read binary message. Message = 1101001.

Step 2: $add_1 = 16$, (from the transition function). This state has two values are $V_{0,1} = 7$ and $V_{1,1} = 9$.

Step 3: The first bit from the message is $m_1 = 1$ select the second value is $V_{1,1} = 9$ because the bit of message is $m_1 = 1$.

Step 4: Encrypt value (9) by McEliece algorithm and use public key matrix (G'): the size of G' matrix = ($k_{mc}\times n_{mc}$). Since, [$\log_2$ value] = 4 bit = k_{mc}, then $C_1 = E(\text{value})_{G'}$. Encrypted the value by McEliece algorithm using the public key (G') and then we add a random error vector (e) length n_{mc} bit and weight(t), the input k_{mc} bit and output n_{mc} bit:

$$\text{value} = 1001 \text{ then } C_1 = \text{value}\times G'+e$$

$$C_1 = [1001]\times \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}_{4\times 7} + [0000100] = [1010010]$$

$C_1 = 82$ is cipher of m_1.

Step 5: $add_2 = 12$, (from the transition function). This state has two values are $V_{0,2} = 5$ and $V_{1,2} = 15$.

Step 6: The first bit from the message: $m_2 = 1$. Select the second value is $V_{1,2} = 15$ because the bit of message is $m_2 = 1$.

Step 7: Encrypt value (15) by McEliece algorithm and use public key matrix (G'): the size of G' matrix = $k_{mc}\times n_{mc}$. Since, [$\log_2$ value] = 4 bit = k_{mc}, then $C_2 = E(\text{value})_{G'}$.

Encrypt the value by McEliece algorithm using the public key (G') and then we add a random error vector (e) length n_{mc} bit and weight(t), the input k_{mc} bit and output n_{mc} bit:

$$\text{value} = 1111 \text{ then } C_2 = \text{value}\times G^{'}+e$$

$$C_2 = [1111]\times \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}_{4\times 7} + [0000100]$$

$C_2 = 123$ is cipher of m_2. Repeat this process to the end of message. Cipher text (C) : 82, 123, 3, 85, 31, 85, 103 (Table 2).

Table 2: Encryption process (ME-RADG-Diffie-Hellman)

Index	Message	State$_{no}$	Status	Jump state	E value	N state$_{no}$
0	1	16	In R	Jump	82	12
1	1	12	In Q		123	11
2	0	11	In Q		3	9
3	1	9	In Q		85	5
4	0	5	In Q		31	10
5	0	20	In R	Jump	85	7
6	1	7	In Q		103	-

Table 3: Decryption process (ME-RADG--Diffie-Hellman)

Index	Ciphertext	Values	State$_{no}$	Status	Jump state	N state$_{no}$	Message
0	82	9	16	In R	Jump	12	1
1	123	15	12	In Q		11	1
2	3	7	11	In Q		9	0
3	85	14	9	In Q		5	1
4	31	11	5	In Q		10	0
5	85	14	20	In R	Jump	7	0
6	103	3	7	In Q		-	1

Table 4: Integrity in ME-RADG-Diffie-Hellman

Index	Ciphertext	Values	State$_{no}$	Status	Jump state	N state$_{no}$	Message
0	82	9	16	In R	Jump	12	1
1	123	15	12	In Q		11	1
2	3	0	11	In Q		-	-
3	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-

Decryption: Decryption the message depends on 2 functions: the decrypt function to decrypt the values and transition function to transition between the states.

Decrypt the first value 82 by using private key of Alice in McEliece algorithm. The result is value (9). Secret key transition of Alice (receiver) = Y_B^{XA} mod q = 11^3 mod 13 = 5 first state = $\propto^{\text{Secret key transition of Alice)}}$ mod q = 2^5 mod 13 = 6 (Table 3).

Since, the address of first the state is 6 and the state 6 is a jump state, then search value 9 in R set find in state 16, the value 6 in state 16 is bit (1) is first bit in original message (Fig. 4).

Decrypt the second value 123 by using private key of Alice in McEliece algorithm. The result is value (15) second state = $\propto^{\text{Secret key transition of Alice+1}}$ mod q = 2^6 mod 13 = 12, state 12 is from Q set, the value 15 in state 12 is bit (1) is second bit in original message. Repeating all the way until the cipher is finished. The message is 1 1 0 1 0 0 1.

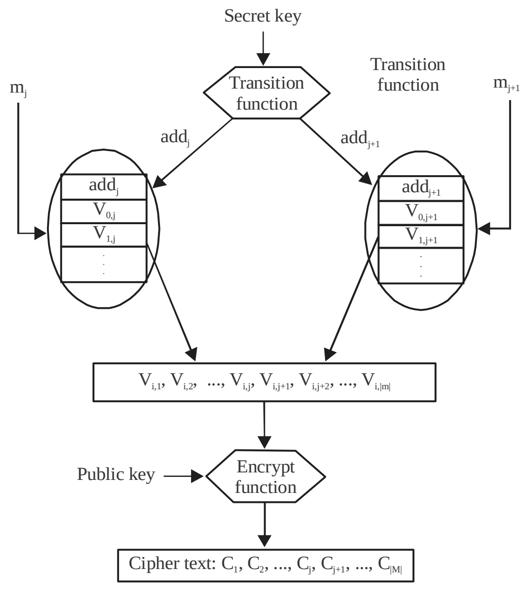

Fig. 4: General design of transition function IN ME-RADG-Diffie-Hellman

Analysis: The performance and security analysis of this method (McEliece RADG using Diffie-Hellman key exchange security system) is explained in E. examples in section (V).

Data integrity: In terms of data and network security, data integrity is the assurance that information can only be accessed or modified by those authorized to do[7]. In this algorithm, any modify on the transferring data by unauthorized users or try to damaging the data by adding noise or deleted some pieces from the data, then the decipher process cannot be resumed as the previous step in Table 4 in the 3th row if the data change from (3) to (2); then the process cannot be resumed to decipher the message as shown in Table 4.

Authentication: Authentication is a technique used to investigate the identity of the party who generated data by

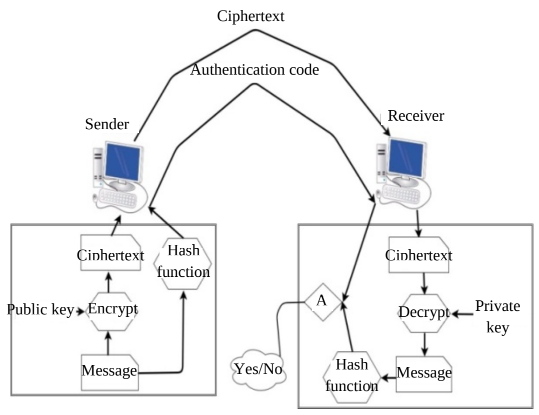

Fig. 5: Authentication in ME-RADG-Diffie-Hellman

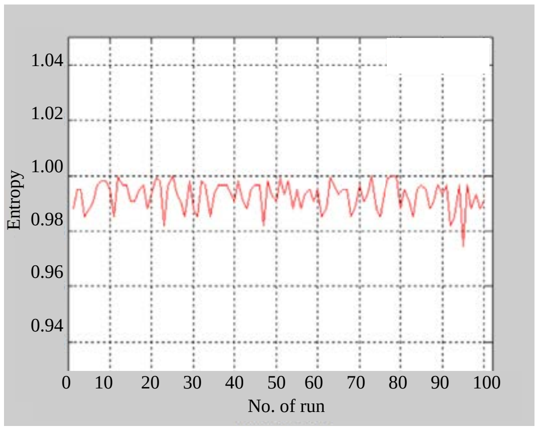

Fig. 6: Entropy for message run 100 times

using hash value. The authentication mechanism joins with the data integrity[8]. In this method using a hash function to provide message authentication and do not need server on network, the whole can be described in three steps (Fig. 5).

Step 1: Each value in massage {V1, V2, ..., V\|M|} encrypt by using the public key and this value are entering into "h" hash function (using one of hashing algorithm like MD5). In MD5 the input length string is 512 bits if PA<512 bits padding with zeros on the left, the sender sends the cipher value and hash value as authentication code.

Step 2: The receiver will decrypt cipher values by using the privet key to get the plaintext, see "D" function in Fig. 3 and 4. The plaintext then compute the hash value like step 1 from the message values (plaintext).

Step 3: The final step is compared between the authentication code and sends from the sender and hash value calculated by receiver if they equal the receiver authentication the sender if the values are not equal the receiver rejects the sender.

Performance analysis of ME-RADG-Diffie-Hellman design: In order to analyze the performance of the ME-RAGE-Diffie-Hellman exchange method, the entropy is used. The ME-RADG- Diffie-Hellman is run 100 times for the same message to show how to implement in terms of entropy of different ciphertext values. In Fig. 6, the entropy for the same text appears 100 times.

CONCLUSION

A new security approach has been proposed based on the classical RADG and the McEliece algorithm. The proposed method, called ME-RADG-Diffie-Hellman, outperforms the original RADG as follows:

The time of decryption is less than the RADG because there is on search process to get the cipher value from all states every time. Now, we don't the state values are generated by the Diffie-Hellman.

The security is upgraded using the asymmetric design (only one person need to privet key). The system design is more dynamic including more parameters in McEliece algorithm.

REFERENCES

01. Kaufman, C., M. Speciner and R. Perlman, 2002. Network Security: Private Communication in a Public World. Prentice Hall, New Jersey, USA.,.
02. Stallings, W., 2010. Cryptography and Network Security: Principles and Practices. 5th Edn., Prentice-Hall, Upper Saddle River, New Jersey, ISBN-10: 0136097049, Pages: 744.
03. Albermany, S.A. and G.A Safdar, 2014. Keyless security in wireless networks. Wirel. Pers. Commun., 79: 1713-1731.
04. McEliece, R.J., 1978. A public-key cryptosystem based on algebraic. National Aeronautics and Space Administration, Washington, USA.
05. Canteaut, A. and N. Sendrier, 1998. Cryptanalysis of the original McEliece cryptosystem. Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, October 18-22, 1998, Springer, Beijing, China, pp: 187-199.
06. Pless, V., 1989. Introduction to the Theory of Error Correcting Codes. John Wiley and Sons Inc., New York, USA.
07. Kou, W., 2012. Networking Security and Standards. Vol. 394, Springer, Berlin, Germany, ISBN:978-1-4613-7820-4, Pages: 203.
08. Boyd, C. and A. Mathuria, 2013. Protocols for Authentication and Key Establishment. Springer, Berlin, Germany, ISBN:978-3-642-07716-6, Pages: 321.