Indian Journal of Science and Technology, Vol 8(15), DOI: 10.17485/ijst/2015/v8i15/64749, July 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Modified ElGamal Elliptic Curve Cryptosystem using Hexadecimal Representation

Ziad E. Dawahdeh[1*], Shahrul N. Yaakob[1] and Ali Makki Sagheer[2]

[1]School of Computer and Communication Engineering, UniMAP University, Malaysia; mziadd@hotmail.com

[2]Information System Department, University of Anbar, Anbar, Iraq

Abstract

Data encryption is an important issue and widely used in recent times to protect the data over internet and ensure security. One of the mostly used in public key cryptographies is the Elliptic Curve Cryptography (ECC). A new modified method has been proposed to encrypt/decrypt data using ECC in this paper. This modification converts each character of the plaintext message to its hexadecimal ASCII value of two digits, then separates the value into two values. After that, the transformation is performed on each value into an affine point on the Elliptic Curve E. This transformation is used to modify ElGamal Elliptic Curve Cryptosystem (EGECC) to encrypt/decrypt the message. In modified method, the number of doubling and adding operations in the encryption process has been reduced. The reduction of this number is a key point in the transformation of each character into an affine point on the EC. In other words, the modified method improved the efficiency of the EGECC algorithm. Moreover, using the hexadecimal ASCII value makes EGECC more secure and complicated to resist the adversaries.

Keywords: Decryption, ElGamal Protocol, Elliptic Curve, Elliptic Curve Cryptography, Encryption, Hexadecimal ASCII

1. Introduction

Elliptic Curve (EC) has been introduced and used for the first time in cryptography by Miller[1] and Koblitz[2]. Elliptic Curve Cryptography (ECC) depends on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP). So, the adversaries are not able to attack ECC and solve ECDLP which is infeasible to be solved and has strength security against all kinds of attacks. For this reason, most of the modern cryptographic systems are established based on the EC[3,4]. ECC can be defined over two types of fields: one is the prime field F_p which is suitable for the software applications and the other is the binary field which is suitable for the hardware applications[5]. ECC has some advantages that make it widely used these days such as small storage capacity, faster computations and reduction of the power consumption[13]. These advantages make ECC is a more suitable to be used in smart cards, wireless communications, portable devices, and e-commerce applications[14]. ECC offers the same security level like RSA and ElGamal algorithms with shorter key length which makes it works with a little amount of memory and low power[11,12]. As a result of these advantages of elliptic curve, several studies have been presented by many researchers. For instance, Williams Stallings in 2011 introduced study about ECC in his book[5]. Hongqiang in 2013 proposed an approach to generate a random number k and sped up computing the scalar multiplication in the encryption and decryption processes[6]. An implementation of ElGamal ECC for encryption and decryption a message is also proposed by Debabrat Boruah in 2014[7]. Meltem Kurt and Tarik Yerlikaya in 2013 presented a modified cryptosystem using hexadecimal to encrypt data. Their study depended on Menezes Vanstone ECC algorithm by adding additional features[8]. Maria Celestin and K. Muneeswaran in 2013 used decimal ASCII value to represent the

*Author for correspondence

characters. These characters are transformed into points on the elliptic curve through multiplying their values by a random point on the Elliptic Curve[9,10].

In this work, a modified method that uses ElGamal ECC for encryption and decryption of the plaintext has been proposed. The modified method uses the hexadecimal ASCII value to represent each character. This representation reduces points doubling and addition which are required to transform the characters into points on the elliptic curve. As a result, further from speeding up the computations can be achieved.

This paper is organized as follows. Section 2 presents a synopsis of the mathematical background to explain elliptic curve E over prime field. Section 3 briefly reviews ECC algorithm and ElGamal protocol. Section 4 explains the modified cryptosystem for encryption and decryption. Section 5 explains a simple example of the proposed method. The comparison between the proposed method and Maria method is discussed in the Section 6. Finally, section 7, displays the conclusion and the advantages of the proposed method.

2. Introduction to Elliptic Curve over Prime Field

Definition 2.1: Let $p \neq 2,3$, an elliptic curve E over a prime field F_p is defined by

$$E: y^2 \equiv x^3 + Ax + B \left(mod\ p\right) \tag{1}$$

where $A, B \in F_p$ and satisfy the condition $4A^3 + 27B^2 \not\equiv 0 \left(mod\ p\right)$. The set of all points (x,y) that satisfy an elliptic curve Equation 1, with a special point O (that is called a point at infinity), forms an elliptic curve group $E(F_p)$[15,16].

2.1 Arithmetic on Elliptic Curve

2.1.1 Point Addition

Suppose $p = (x_1, y_1)$ and $Q = (x_2, y_2)$, where $P \neq Q$, are two points lie on an elliptic curve E defined in Equation 1. The sum $P + Q$ results a third point R which is also lies on E. To add two points on E there are some cases on the coordinates of the points P and Q. These cases are given as follows[8]:

- If $p \neq Q \neq O$ with $x_1 \neq x_2$. Then sum of P and Q in this case is defined by

$$P + Q = R = (x_3, y_3) \tag{2}$$

where

$$\lambda = \frac{\left(y_2 - y_1\right)}{\left(x_2 - x_1\right)} \tag{3}$$

$$x_3 \equiv (\lambda^2 - x_1 - x_2)(mod\ p) \tag{4}$$

$$y_3 \equiv (\lambda(x_1 - x_3) - y_1)(mod\ p) \tag{5}$$

- If $x_1 = x_2$ but $y_1 \neq y_2$ then $P + Q = O$.

2.1.2 Point Doubling

Let $p = (x_1, y_1)$ be a point lies on E. Adding the point P to itself is called doubling point on an elliptic curve E[17,18]. In other words

$$P + P = 2P = R = (x_3, y_3) \tag{6}$$

where

$$\lambda = \frac{3x_1^{\ 2} + A}{2y_1} \tag{7}$$

$$x_3 \equiv (\lambda^2 - 2x_1)(mod\ p) \tag{8}$$

$$y_3 \equiv (\lambda(x_1 - x_3) - y_1)(mod\ p) \tag{9}$$

2.2 Scalar Multiplication (Point Multiplication)

Suppose k is an integer and $p = (x_1, y_1)$ is a point lies on E. The scalar multiplication can be defined by

$$kP = \underbrace{P + P + \cdots + P}_{k-times}, \tag{10}$$

In other words, adding a point P to itself K times[17].

A scalar multiplication kP can be computed using the point doubling and point addition laws. For example, the

scalar multiplication $9P$ can be calculated by the following expression:

$$9P = 2(2(2P)) + P.$$

3. ElGamal Elliptic Curve Cryptosystem

In 1987, Koblitz proposed analogues of ElGamal public key cryptosystem based on the elliptic curve over a finite prime field. The first step of ElGamal elliptic curve cryptosystem converts the plaintext message m to a point P_m on the elliptic curve $E(F_p)$. Each party chooses a private key randomly from the interval $[1, p-1]$; n_A for user A and n_B for user B, then computes a public key by multiplying the private key by the base point G ($P_A = n_A.G$ and $P_B = n_B.G$). To encrypt the message P_m; the sender chooses a random number k and multiplies it by the receiver public key P_B then adds the result to the message P_m and send it with kG So, the ciphertext message will be $\{k.G, P_m + k.P_B\} = \{(x_1, y_1), (x_2, y_2)\}$. To decrypt the ciphertext, the receiver multiplies kG by his private key n_B and subtracts the result from $P_m + k.P_B$ to get the plaintext P_m as follows[9,15]:

$$P_m + k.P_B - n_B(k.G) = P_m + k.P_B - k(n_B.G) = P_m + k.P_B - k.P_B = P_m$$

4. The Modified Cryptosystem

The Modification of ElGamal Elliptic Curve Cryptosystem (MEGECC) has been presented in this section. This modification depends on the speeding up of the computation on EGECC using hexadecimal ASCII values by reducing the number of doubling and addition operations needed. The domain parameters (that is $\{A,B,p,G\}$) are public for all entities. Suppose A and B are two users wishing to communicate and exchange the information using MEGECC over insecure channel. Let us choose the user A as the sender who wants to encrypt and send a message m to the user B (the receiver). Every entity, namely A and B, need to choose a private key. The private keys, n_A and n_B are positive integers chosen randomly from the interval $[1, p-1]$. The public keys for the users A and B can be generated respectively as follows:

$$P_A = n_A.G$$
$$P_B = n_B.G$$

The basic idea of the contribution in this work depends on using the hexadecimal ASCII value to reduce the number of doubling and addition operations. Suppose user A wants to send a message m to user B. Firstly, he converts each character in the message m into hexadecimal ASCII value of two digits $(h_1\ h_2)_{16}$ then separates the value into two values $(h_1, h_2)_{16}$ and converts each value of h_1 and h_2 to decimal values d_1 and d_2 respectively[8]. The scalar multiplication of the base point G on E by each value of d_1 and d_2 can be computed to transform the values to points on E by the following formulas:

$$P_{h_1} = d_1.G$$
$$P_{h_2} = d_2.G$$

where P_{h_1} and P_{h_2} are two points lie on E.

User A computes the secret key K by multiplying his private key n_A by B's public key P_B

$K = n_A P_B$

and adds the result to the points P_{h_1} and P_{h_2} to compute the ciphertext message as follows:

$$C_1 = P_{h_1} + K$$

$$C_2 = P_{h_2} + K$$

where $C1$ and $C2$ are two points lie on E. The set of points $\{C1,C2\}$ is sent to the user B.

Upon receiving the ciphertext $\{C1,C2\}$ by user B, the decryption process will be started. User B first needs to multiply his private key n_B by A's public key P_A to get the secret key K

$K = n_B P_A$ then subtracts K from $C1$ and $C2$ to get P_{h_1} and P_{h_2}

$$C_1 - K = C_1 - n_B.P_A$$
$$= P_{h_1} + n_A.P_B - n_B.n_A.G$$
$$= P_{h_1} + n_A.n_B.G - n_A.n_B.G$$

$= P_{h_1}$ and in similar way for C2

$$C_2 - K = C_2 - n_B.P_A$$
$$= P_{h_2} + n_A.P_B - n_B.n_A.G$$
$$= P_{h2} + n_A.n_B.G - n_A.n_B.G$$

$= P_{h_2}$

Next step is to solve the following equations for d_1 and d_2 by using Elliptic Curve Discrete Logarithm Problem (ECDLP) where P_{h_1}, P_{h_2}, and G are known.

$$P_{h_1} = d_1.G$$
$$P_{h_2} = d_2.G$$

The last step is to convert d_1 and d_2 to hexadecimal h_1 and h_2 respectively, and write them as, $(h_1\ h_2)_{16}$ then find the match character from the hexadecimal ASCII table. Repeat the previous procedure for each character in the message m.

One of the advantages of the modified cryptosystem is that the solution of $P_{h_1} = d_1.G$ and $P_{h_2} = d_2.G$ is not difficult for the receiver and will not take a long time because the largest value for d_1 and d_2 in decimal is 15 (the maximum digit in hexadecimal is F = 15) but it is very difficult for the adversary because he can't know the private key n_B and the prime number p will be chosen as a large number.

4.1 The Proposed Algorithm (MEGECC)

4.1.1 Step 1: Key Scheduling

4.1.1.1 User A

- Choose the private key $n_A \in [1, p-1]$.
- Compute the public key $P_A = n_A.G$.

4.1.1.2 User B

- Choose the private key $n_B \in [1, p-1]$.
- Compute the public key $P_B = n_B.G$.

The secret key will be $K = n_A.P_B = n_B.P_A = n_A.n_B.G$

4.1.2 Step 2: Encryption

4.1.2.1 User A

- Convert each character of the message m to hexadecimal ASCII value of two digits $(h_1\ h_2)_{16}$.
- Rewrite the value $(h_1\ h_2)_{16}$ as $(h_1, h_2)_{16}$ then convert it to two decimal values $(d_1, d_2)_{10}$
- Compute $P_{h_1} = d_1.G$ // G is the base point.
- Compute $P_{h_2} = d_2.G$ // to represent the values d_1 and d_2 as points on $E(Fp)$.
- Compute $K = n_A.P_B$ // n_A the private key of user A and P_B the public key of user B.
- Compute $C_1 = P_{h_1} + K$
- Compute $C_2 = P_{h_2} + K$
- Send $[\{C]_1, C_2\}$ to user B.

4.1.3 Step 3: Decryption

4.1.3.1 User B

- Compute $K = n_B.P_A$.
- Compute $P_{h_1} = C_1 - K$.
- Compute $P_{h_2} = C_2 - K$.
- Extract d_1 from p_{h_1}.// by solving the discrete logarithm problem $P_{h_1} = d_1.G$
- Extract d_2 from p_{h_2}.//. by solving the discrete logarithm problem $P_{h_2} = d_2.G$
- Convert $(d_1, d_2)_{10}$ to hexadecimal $(h_1, h_2)_{16}$ and rewrite it as $(h_1\ h_2)_{16}$.
- Find the match character for $(h_1\ h_2)_{16}$ from the hexadecimal ASCII table.

5. Implementation Example

Assume that user A and user B are agreed to use the elliptic curve

$$E: y^2 \equiv x^3 + x + 3\ (mod\ 31)$$

where $A = 1, B = 3, p = 31$ satisfy the condition $4A^3 + 27B^2$ = $4(1)^3+27(3)^2$ = 4+243 = 247 mod 31 = 30 ≠ 0, then the points of the elliptic curve $E_{31}(1,3)$ are shown in Table 1.

Let the point (1,6) be chosen as the base point G; the order of our elliptic curve is 41 and it is a prime

Table 1. Points on the elliptic curve $E: y^2 \equiv x^3+x+3 (mod\ 31)$

(1, 6)	(1, 25)	(3, 8)	(3, 23)	(4, 3)	(4, 28)	(5, 3)	(5, 28)
(6, 15)	(6, 16)	(9, 11)	(9, 20)	(12, 10)	(12, 21)	(14, 8)	(14, 23)
(15, 13)	(15, 18)	(17, 2)	(17, 29)	(18, 5)	(18, 26)	(20, 5)	(20, 26)
(21, 4)	(21, 27)	(22, 3)	(22, 28)	(23, 14)	(23, 17)	(24, 5)	(24, 26)
(26, 11)	(26, 20)	(27, 11)	(27, 20)	(28, 2)	(28, 29)	(30, 1)	(30, 30)

number, so we can choose any point as the base point or generator[19]. So, the domain parameters are $\{A, B, p, G\} = \{1, 3, 31, (1,6)\}$. If user A wants to send the message '**Hello**' to user B, he should first convert each character in the message '**Hello**' to the hexadecimal value from the ASCII table, then separates each value into two values and converts them to decimal values.

$H \rightarrow (48)_{16} \rightarrow (4, 8)_{16} \rightarrow (4, 8)_{10}$
$e \rightarrow (65)_{16} \rightarrow (6, 5)_{16} \rightarrow (6, 5)_{10}$
$l \rightarrow (6C)_{16} \rightarrow (6, C)_{16} \rightarrow (6, 12)_{10}$
$l \rightarrow (6C)_{16} \rightarrow (6, C)_{16} \rightarrow (6, 12)_{10}$
$o \rightarrow (6F)_{16} \rightarrow (6, F)_{16} \rightarrow (6, 15)_{10}$

Now, apply the proposed algorithm on the first character "**H**". If user A wants to send the character "**H**" to user B, he should first convert the character "**H**" to the hexadecimal value from the ASCII table $(48)_{16}$, then separate the value into two values $(4, 8)_{16}$ and convert it to decimal values $(4, 8)_{10}$, then do the following calculations:

5.1 Step 1: Key Scheduling

5.1.1 User A

- Choose the private key $n_A = 13 \in [1, 30]$.
- Compute the public key $P_A = n_A.G = 13(1,6) = (3, 23)$.

5.1.2 User B

- Choose the private key $n_B = 17 \in [1, 30]$.
- Compute the public key $P_B = n_B.G = 17(1,6) = (24,5)$. P_A and P_B will be exchanged and be public for both users A and B.

5.2 Step 2: Encryption

5.2.1 User A

- "H"$\rightarrow (48)_{16}$
- $(48)_{16} \rightarrow (4,8)_{16} \rightarrow (4,8)_{10}$
- $P_{h_1} = d_1.G = 4(1,6) = (23,17)$
- $P_{h_2} = d_2.G = 8(1,6) = (18,5)$
- $K = n_A.P_B = 13(24,5) = (20,5)$
- $C_1 = P_{h_1} + K = (23,17) + (20,5) = (4,28)$
- $C_1 = P_{h2} + K = (18,5) + (20,5) = (24,26)$
- Send $\{(4,28),(24,26)\}$ to user B.

5.3 Step 3: Decryption

5.3.1 User B

- $K = n_B.P_A = 17(3,23) = (20,5)$
- $P_{h_1} = C_1 - K = (4,28) - (20,5) = (4,28) + (20,-5) = (4,28) + (20,26) = (23,17)$
- $P_{h_2} = C_2 - K = (24,26) - (20,5) = (24,26) + (20,-5) = (24,26) + (20,26) = (18,5)$
- Extract d_1=4 from p_{h_1}.// by solving the discrete logarithm problem (23,17)= $d_1(1,6)$
- Extract d_2=8 from p_{h_2}.// by solving the discrete logarithm problem (18,5)= $d_2(1,6)$
- Convert $(4,8)_{10}$ to hexadecimal $(4,8)_{16}$ and rewrite it as $(48)_{16}$.
- Find the match character for $(48)_{16}$ from the hexadecimal ASCII table which is "**H**".

In the proposed algorithm, solving $P_{h_1} = d_1.G$ and $P_{h_2} = d_2.G$ for d_1 and d_2 is needed, it is not difficult and will not take a long time because the largest value for d_1 and d_2 in decimal is 15 (the maximum digit in hexadecimal is F = 15). The same processes for the other characters "ello" should be repeated.

6. Results and Discussions

In this section, a comparison between the proposed method in this paper and the proposed method by Maria Celestin and K. Muneeswaran[9] is done on the plaintext "**Hello**" with consider of doubling operation ($2G = G + G$) is same as addition operation (G+Q). Take the character "**H**" as an example; in the proposed method the following operations are required

$H \rightarrow (48)_{16} \rightarrow (4,8)_{16} \rightarrow (4,8)_{10}$

then calculate $4G$=2(2G) and $8G$=2(2(2G)), so the total operations are 2D+3D=5 operations. Whereas, in Maria method '**H**'$\rightarrow (72)_{ASCII}$, then calculate 72G = 2(2(2(2(2G))+G))) and the total operations are 6D+1A=7 operations (D for doubling and A for addition). So, in the proposed method the character "**H**" needs 5 operations where in Maria Method it needs 7 operations. Table 2 summaries the operations that are required for each method to transform the plaintext "**Hello**" into affine points on the EC.
Table 2 shows that the proposed method is better than

Table 2. The required doubling and addition operations for the plaintext "Hello"

The character / The method	H	e	l	l	o	Total operations
The proposed method	5D	4D + 2A	5D + 2A	5D + 2A	5D + 4A	34
Maria method	6D + 1A	6D + 3A	6D + 3A	6D + 3A	6D + 5A	45

Maria method. In this method, to transform the character "**H**" which has the hexadecimal ASCII value 48 into an affine point on the EC we need 5 operations. Whereas, in Maria method the decimal ASCII value for "**H**" is 72, so the sender needs 7 operations to do the transformation. The total operations that is needed for the plaintext "**Hello**" is 34 in the proposed method and 45 in Maria method and the difference between the two methods will be increased if the size of the plaintext is increased.

Figure 1 represents a column chart graph for the arithmetic operations (doubling and adding) that are required in both methods for the plaintext "**Hello**".

Table 3 shows the number of doubling and addition operations that are needed to transform plaintexts of different sizes into affine points on the EC in the proposed method and Maria method and the percentage of improvement.

To calculate the improvement percentage for the plaintext "**Hello**", subtracts number of operations in the proposed method from number of operations in Maria method and divide by number of operations in Maria method then multiply by 100% as follows:

Improvement percentage for "Hello" =

$$\frac{45-34}{45} \times 100\% = 24.44\%$$

It is clear from Table 3 that the proposed method is better than Maria method and the percentage of improvement increases when the plaintext size increases.

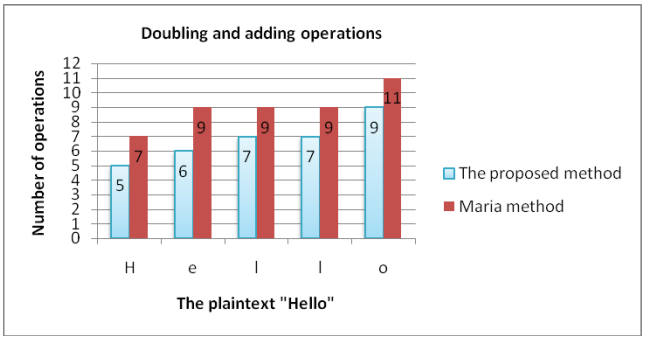

Figure 1. Column chart for doubling and adding operations for the plaintext "**Hello**".

7. Conclusions

Elliptic Curve Cryptosystem (ECC) is one of the most efficient cryptosystems that is used to encrypt/decrypt data; it is secured against all kinds of attacks. The short key size of ECC gives it strengthened security compared to other cryptosystems like RSA with the same security level. This advantage leads to fast computations, less memory and power consumption, and saving bandwidth. These advantages make ECC efficient to be used in some applications like e-commerce, smart cards, chip cards, and portable devices.

In our work, a new efficient method has been proposed to encrypt/decrypt any text using the hexadecimal ASCII value for each character. The main contribution is to reduce the number of doubling and addition operations as shown in Table 2 and Table 3 that user A needs to transform the plaintext into an affine points on the EC

Table 3. The required operations and improvement percentage for different plaintexts

The plaintext	The size(number of characters)	Number of doubling and addition operations		Improvement percentage
		The proposed method	Maria method	
"**Hello**"	5	34	45	24.44 %
"Conclusion"	10	70	97	27.83 %
"Implementations"	15	99	139	28.78 %
"Scalar Multiplication"	20	119	180	33.88 %

=and speed up the encryption process.

In the modified method, user B can solve the small values of the discrete logarithm problems $P_{h_1} = d_1.G$ and $P_{h_2} = d_2.G$ easily. These computations not need long time because the largest value for d_1 and d_2 in decimal is 15 that corresponds to the maximum digit in hexadecimal F = 15, but at the same time it is very difficult for the adversary because it is difficult for him to know n_A and n_B. Moreover, in the modified method, user A does not need to send kG every time, he sends the ciphertext message only, because both users A and B working on their private keys n_A and n_B. Separating each character of the plaintext into two points on the EC and using the hexadecimal ASCII value for the transformation make modified method more secure and difficult for the adversaries.

8. References

1. Miller VS. Use of elliptic curves in cryptography. Advances in Cryptology. Proceedings of Crypto85. Lecture note in Computer Science. Berlin, Heidelberg: Springer-Verlag; 1986. p. 417–26.
2. Koblitz N. Elliptic curve cryptosystems. Mathematics *of* Computation. 1987; 48:203–9.
3. Sagheer AM. Elliptic curves cryptographic techniques. Proceeding of the 6th International Conference on Signal Processing and Communication Systems (ICSPCS'2012); Gold Coast, Australia: IEEE; 2012.
4. Erich W, Paul W. Solving the Discrete Logarithm of a 113-bit Koblitz Curve with an FPGA Cluster. Selected Areas in Cryptography-SAC. Springer International Publishing; 2014. p. 363–79.
5. William S. Cryptography and Network Security: Principles and Practices. 5th ed. Prentice Hall; 2011.
6. Lv H, Li H, Yi J, Lu H. Optimal implementation of elliptic curve cryptography. IEEE International Conference on Service Operations and Logistics and Informatics; Dongguan, Guangdong, China. 2013 Jul 28–30.
7. Boruah D, Saikia M. Implementation of ElGamal Elliptic Curve Cryptography over prime field using C. IEEE International Conference on Information Communication and Embedded Systems (ICICES); 2014.
8. Kurt M, Yerlikaya Y. A new modified cryptosystem based on Menezes Vanstone elliptic curve cryptography algorithm that uses characters' hexadecimal values. International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE); Konya: IEEE; 2013 May 9–11. p. 449–53.
9. Vigila SMC, Muneeswaran K. A new elliptic curve cryptosystem for securing sensitive data applications. International Journal of Electronic Security and Digital Forensics. 2013; 5(1):11–24.
10. Vigila S, Muneeswaran K. Implementation of text based cryptosystem using elliptic curve cryptography. IEEE Proceedings of Advanced Computing Conference; 2009.
11. Alese BK, Philemon ED, Falaki SO. Comparative analysis of public-key encryption schemes. IJET. 2012 Sep; 2(9):1552–68.
12. Rajadurga K, Kumar SR. GF (2m) based low complexity multiplier for elliptic curve cryptography systems. Networking and Communication Engineering. 2014; 6(4): 150–5.
13. Fu M, Wei C. Elliptic curve cryptosystem ElGamal encryption and transmission scheme. International Conference on Computer Application and System Modeling (ICCASM 2010); Taiyuan: IEEE; 2010 Oct 22–24. p. 51–3.
14. Gupta K, Silakari S. ECC over RSA for asymmetric encryption: A review. International Journal of Computer Science Issues. 2011 May; 8(3):370–5.
15. Hankerson D**,** Menezes A, Vanstone S. Guide to elliptic curve cryptography. New York: Springer Science and Business Media; 2004.
16. Hoffstein J, Pipher JC, Silverman JH. Elliptic curves and cryptography. An Introduction to Mathematical Cryptography. New York: Springer; 2014. p. 299–371.
17. Biswojit N. Signcryption schemes based on elliptic curve cryptography [Master Thesis]. Rourkela, India: National Institute of Technology; 2014.
18. Udin MN, Halim SA, Jayes MI, Kamarulhaili H. Application of message embedding technique in ElGamal elliptic curve cryptosystem. International Conference on Statistics in Science, Business, and Engineering (ICSSBE); Langkawi: IEEE; 2012 Sep 10–12. p. 1–6.
19. Sagheer AM. Enhancement of elliptic curves cryptography methods [MSc Thesis]. Baghdad, Iraq: University of Technology; 2004