Modified ElGamal Elliptic Curve Cryptosystem using Hexadecimal Representation

Ziad E. Dawahdeh1*, Shahrul N. Yaakob1 and Ali Makki Sagheer2

1School of Computer and Communication Engineering, UniMAP University, Malaysia; mziadd@hotmail.com
2Information 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 Miller1 and Koblitz2. 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 EC3,4. 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 applications5. ECC has some advantages that make it widely used these days such as small storage capacity, faster computations and reduction of the power consumption13. These advantages make ECC is a more suitable to be used in smart cards, wireless communications, portable devices, and e-commerce applications14. 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 power11,12. 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 book5. Hongqiang in 2013 proposed an approach to generate a random number \( k \) and sped up computing the scalar multiplication in the encryption and decryption processes6. An implementation of ElGamal ECC for encryption and decryption a message is also proposed by Debabrat Boruah in 20147. 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 features8. Maria Celestin and K. Muneeswaran in 2013 used decimal ASCII value to represent the
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

\[
E : y^2 = x^3 + Ax + B \pmod p
\]

where \(A, B \in F_p\) and satisfy the condition \(4A^3 + 27B^2 \neq 0 \pmod p\). 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

\[
P + Q = R = (x_3, y_3)
\]

where

\[
\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)} \quad (3)
\]

\[
x_3 = (\lambda^2 - x_1 - x_2)(mod p)
\]

\[
y_3 = (\lambda(x_1 - x_3) - y_1)(mod p)
\]

- 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

\[
P + P = 2P = R = (x_3, y_3)
\]

where

\[
\lambda = \frac{3x_1^2 + A}{2y_1} \quad (7)
\]

\[
x_3 = (\lambda^2 - 2x_1)(mod p)
\]

\[
y_3 = (\lambda(x_1 - x_3) - y_1)(mod p)
\]

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

\[
kP = P + P + \cdots + P \quad (k-times)
\]

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 $G(P_a = n_AG$ and $P_B = n_BG$). 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 $\{kG, P_m + kP_B\} = (x_1, y_1), (x_2, y_2)$. To decrypt the ciphertext, the receiver multiplies $kG$ by his private key $n_B$ and subtracts the result from $P_m + kP_B$ to get the plaintext $P_m$ as follows:

$$P_m + kP_B - n_B(kG) = P_m + kP_B - k(n_BG) = P_m + kP_B - kP_B = P_m$$

### 4. The Modified Cryptosystem

The Modification of ElGamal Elliptic Curve Cryptosystem (MEGEC) 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 MEGEC 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:

$$P_A = n_AG$$

$$P_B = n_BG$$

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 ($h_1, h_2$) then separates the value into two values ($h_1, h_2$) and converts each value of $h_1$ and $h_2$ to decimal values $d_1$ and $d_2$ respectively. The scalar multiplication of the base point $G$ on each character in the message $m$ by the value of $d_1$ and $d_2$ can be computed to transform the values to points on $E$ by the following formulas:

$$P_{h_1} = d_1G$$

$$P_{h_2} = d_2G$$

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 $P_B$:

$$K = n_A \cdot P_B$$

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 $C_1$ and $C_2$ are two points lie on $E$. The set of points $\{C_1, C_2\}$ is sent to the user B.

Upon receiving the ciphertext $\{C_1, C_2\}$ 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 $K$:

$$K = n_B \cdot P_A$$

then subtracts $K$ from $C_1$ and $C_2$ to get $P_{h_1}$ and $P_{h_2}$:

$$C_1 - K = C_1 - n_BP_A$$

$$= P_{h_1} + n_AP_B - n_Bn_AG$$

$$= P_{h_1} + n_A\cdot n_BG - n_A\cdot n_BG$$

$$= P_{h_1}$$

and in similar way for $C_2$

$$C_2 - K = C_2 - n_BP_A$$

$$= P_{h_2} + n_AP_B - n_Bn_AG$$

$$= P_{h_2} + n_A\cdot n_BG - n_A\cdot n_BG$$

$$= P_{h_2}$$
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 $K = n_B.P_B / / n_B$ 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 $4A^3 + 27B^2 = 4(1)^3 + 27(3)^2 = 4 + 243 = 247 \ mod\ 31 = 30 \neq 0$, 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>
<thead>
<tr>
<th>1, 6</th>
<th>(1, 25)</th>
<th>(3, 8)</th>
<th>(3, 23)</th>
<th>(4, 3)</th>
<th>(4, 28)</th>
<th>(5, 3)</th>
<th>(5, 28)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6, 15</td>
<td>(6, 16)</td>
<td>(9, 11)</td>
<td>(9, 20)</td>
<td>(12, 10)</td>
<td>(12, 21)</td>
<td>(14, 8)</td>
<td>(14, 23)</td>
</tr>
<tr>
<td>15, 13</td>
<td>(15, 18)</td>
<td>(17, 2)</td>
<td>(17, 29)</td>
<td>(18, 5)</td>
<td>(18, 26)</td>
<td>(20, 5)</td>
<td>(20, 26)</td>
</tr>
<tr>
<td>21, 4</td>
<td>(21, 27)</td>
<td>(22, 3)</td>
<td>(22, 28)</td>
<td>(23, 14)</td>
<td>(23, 17)</td>
<td>(24, 5)</td>
<td>(24, 26)</td>
</tr>
<tr>
<td>26, 11</td>
<td>(26, 20)</td>
<td>(27, 11)</td>
<td>(27, 20)</td>
<td>(28, 2)</td>
<td>(28, 29)</td>
<td>(30, 1)</td>
<td>(30, 30)</td>
</tr>
</tbody>
</table>
number, so we can choose any point as the base point or generator\(^9\). 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.

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

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)\(_{10}\) 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 \(P_A = n_A \cdot G = 13(1, 6) = (3, 23)\).

#### 5.1.2 User B

- Choose the private key \(n_B = 17 \in [1, 30]\)
- Compute the public key \(P_B = n_B \cdot G = 17(1, 6) = (24, 5)\). \(P_A\) 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" \(\rightarrow (48)_{16}\)
- \((48)_{16} \rightarrow (4, 8)_{10} \rightarrow (4, 8)_{10}\)
- \(P_h = d_1 \cdot G = 4(1, 6) = (23, 17)\)
- \(P_b = d_2 \cdot G = 8(1, 6) = (18, 5)\)
- \(K = n_A \cdot P_b = 13(24, 5) = (20, 5)\)
- \(C_1 = P_h + K = (23, 17) + (20, 5) = (4, 28)\)
- \(C_1 = P_{h2} + K = (18, 5) + (20, 5) = (24, 26)\)
- Send \(\{(4, 28), (24, 26)\}\) to user B.

### 5.3 Step 3: Decryption

#### 5.3.1 User B

- \(K = n_b \cdot P_A = 17(3, 23) = (20, 5)\)
- \(P_{h_b} = C_1 - K = (4, 28) - (20, 5) = (4, 28) + (20, -5) = (4, 28) + (20, 26) = (23, 17)\)
- \(P_{b_b} = C_2 - K = (24, 26) - (20, 5) = (24, 26) + (20, -5) = (24, 26) + (20, 26) = (18, 5)\)
- Extract \(d_1 = 4\) from \(p_{h_b} \cdot G\) by solving the discrete logarithm problem \((23, 17) = d_1(1, 6)\)
- Extract \(d_2 = 8\) from \(p_{b_b} \cdot G\) 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 \(p_{h_b} = d_1 \cdot G\) and \(p_{b_b} = d_2 \cdot G\) 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

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

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 \(72G = 2(2(2(2(2(2G)) + G)))\) 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...
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.

<table>
<thead>
<tr>
<th>The plaintext</th>
<th>The size(number of characters)</th>
<th>Number of doubling and addition operations</th>
<th>Improvement percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>The proposed method</td>
<td>Maria method</td>
</tr>
<tr>
<td>“Hello”</td>
<td>5</td>
<td>34</td>
<td>45</td>
</tr>
<tr>
<td>“Conclusion”</td>
<td>10</td>
<td>70</td>
<td>97</td>
</tr>
<tr>
<td>“Implementations”</td>
<td>15</td>
<td>99</td>
<td>139</td>
</tr>
<tr>
<td>“Scalar Multiplication”</td>
<td>20</td>
<td>119</td>
<td>180</td>
</tr>
</tbody>
</table>
and speed up the encryption process.

In the modified method, user B can solve the small values of the discrete logarithm problems $P_{d_1} = d_1G$ and $P_{d_2} = d_2G$ 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
