ECC Encryption and Decryption with a Data Sequence

F. Amounas and E. H. El Kinani

1 R.O.I Group, Moulay Ismaïl University Informatics Department
Faculty of Sciences and Technics
Box 509 Errachidia, Morocco

2 A.A Group, Moulay Ismaïl University Mathematical Department
Faculty of Sciences and Technics
Box 509 Errachidia, Morocco

Abstract

The paper describes the basic idea of elliptic curve cryptography (ECC) and its implementation by applying a data sequence on ECC encrypted message over the finite field \( GF(p) \). We also propose an algorithm for to generate a data sequence and use it to the output of our cryptosystem. In ECC, we normally start with an affine point noted \( P_m \). This point lies on the elliptic curve. Here, we will illustrate encryption/decryption involving the embedding system \( m \rightarrow P_m \), which imbed the characters constituting the message and then subjecting it to algorithm proposed. Thus by using a data sequence, the present algorithm provides sufficient strength against crypto analysis and whose performance can be compared with standard algorithms like Rivest-Shamir-Adleman algorithm (RSA).

Keywords: Elliptic Curve Cryptography, Discrete Logarithm, Cryptosystem, Public Key, Data Sequence, Encryption, Decryption

1 Introduction

The idea of using ECC was introduced by Victor Miller [1] and Neal Kolbitz as an alternative to established public key systems such as RSA [2]. In 1985, they proposed a public key cryptosystems analogue of ElGamal encryption schema witch used Elliptic Curve Discrete Logarithm Problem (ECDLP) [3]. The application of elliptic curves to the field of cryptography has been relatively recent. In the paper [4], the authors presented the implementation of ECC by first transforming the message into an affine point on the elliptic
curve(EC), and then applying the knapsack algorithm on ECC encrypted message over the finite field $GF(p)$. The knapsack problem is not secure in the present standards and more over in the work the authors in their decryption process used elliptic curve discrete logarithm to get back the plain text. This may form a computationally infeasible problem if the values are large enough in generating the plain text. In our previous work [5], we provide an example of the public-key cryptosystems based on ECC mechanism. In fact, the transformation of the message into an affine point is explained. A transformed character is encrypted by ECC technique. In the present work, the output of ECC algorithm is provided with a values of the sequence generated by the proposed algorithm. More precisely, this paper presents an algorithm to generate a data sequence and its application to the output of our cryptosystem. We illustrate also the implementation of our cryptosystem based into an elliptic curve caracterised by the following equation: $y^2 = x^3 - x + 16$[29].

2 Overview of Elliptic curve Cryptosystem

Elliptic curve cryptosystems (ECCs) include key distribution, encryption algorithms. The key distribution algorithm is used to share a secret key and the encryption algorithm enables confidential communication. ECCs are based on the addition of rational points on a chosen elliptic curve. An elliptic curve $E$ over the finite field $GF(p)$ where $p$ is a prime, is the set of points $(x, y)$ satisfying the following equation:

$$E : y^2 = x^3 + ax + b,$$

(1)

where $a, b$ are integer modulo $p$, satisfying: $4a^3 + 27b \neq 0 \mod p$, and include an point $\Omega$ called point at infinity.

It is know that rational points form an additive group in the addition over the elliptic curve shown in the following figure:

![Figure 1 Addition rule over an elliptic curve](image-url)
When points P and Q on the elliptic curve E shown in Figure.1 are added, the result is defined as the point S obtained by inverting the sign of the y-coordinate of point R, where R is the intersection of E and the line passing through P and Q. If P and Q are at the same position, the line is the tangent of E at P. Moreover, the sum of the point at infinity and a point P is defined as point P itself.

3 Proposed Method Description

The proposed algorithm requires that we generate a serie of vectors called $S_i$. The procedure is show in algorithm (1). Then, the result of this procedure is applied to the output of ECC.

3.1 Algorithm (1) for generating the sequence

1. Let P is a point generator and $n$ is order of P

2. Consider the sequence for $i=0$ to $n$ values

3. Convert each element of the sequence in base 3, and let $m$ be a number of digits: For example: $n = 34 \implies$ we obtain $m=4$


4. Represent above form in $(n+1)\times m$ matrix (M)

$$M = \begin{pmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,m-1} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,m-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,0} & a_{n,1} & \cdots & a_{n,m-1} \end{pmatrix}$$

5. Circularly shifting each row of M by one element to the right

$$[a_{i,0} \ a_{i,1} \ a_{i,2} \ a_{i,m-1}] \implies [a_{i,m-1} \ a_{i,0} \ a_{i,1} \ a_{i,2} \ a_{i,m-2}]$$

6. The sequence formed is:

$$S : [S_0 = [a_{0,m-1} \ a_{0,0} \ a_{0,1} \ a_{0,2} \ \cdots \ a_{0,m-2}], S_1 = [a_{1,m-1} \ a_{1,0} \ a_{1,1} \ a_{1,2} \ \cdots \ a_{1,m-2}], \ldots, S_n = [a_{n,m-1} \ a_{n,0} \ a_{n,1} \ a_{n,2} \ \cdots \ a_{n,m-2}]]$$
3.2 Algorithm (2) ECC by using a sequence generated

Suppose that we have some elliptic curve $E$ defined over a finite field $GF(p)$ and that $E$ and a point $P \in E$ are publicly known, as is the embedding system $m \rightarrow P_m$; which imbed plain text on an elliptic curve $E$. Then, when Alice wants to communicate secretly with Bob, they proceed in the following way:

-Encryption

**Step 1.** Bob chooses a random integer $a$, and publishes the point $aP$ (while $a$ remains secret).

**Step 2.** Alice chooses her own random integer $l$ and computes the pair of points:

$$P_1(x_1, y_1) = lP$$
$$P_2(x_2, y_2) = P_i + l(aP)$$

**Step 3.** Read the sequence generated from algorithm (1).

**Step 4.** Calculate $S(x_1, y_1)$ and $S(x_2, y_2)$ with $S$ is a corresponding sequence value in step3. Then, the ciphertext is as following:

$$C_m = (S(x_1, y_1), S(x_2, y_2))$$

**Step 5.** Alice converts $C_m$ to binary form with: $0 \rightarrow 00$, $1 \rightarrow 01$, $2 \rightarrow 10$ and send to Bob a serie of bits.

- Decryption

To decrypt the message, Bob knows the sequence of $S_i$, his own secret $a$, the base point $P,a,b,p$ values of the ECC. Bob receives the encrypted message $C_m = (S(x_1, y_1), S(x_2, y_2))$

**Step 1.** Converts a binary form to serie of digits as well as: $00 \rightarrow 0$, $01 \rightarrow 1$, $10 \rightarrow 2$

**Step 2.** Transforms $C_m$ into groups of $2m$.

**Step 3.** Extract a group of $m$ digits in sequence of step2.

**Step 4.** Circularly shifting this sequence of $m$ digits by one element to the left

**Step 5.** Convert a sequence to decimal form, and store a value in $k$

For example: $0100$ will in the form: $0 \cdot 3^3 + 1 \cdot 3^2 + 0 \cdot 3^1 + 0 \cdot 3^0 = 9$

$\rightarrow k=9$

**Step 6.** Obtain $(k+1)P$ from pre-computed and stored point $(k+1)P=(x_1, y_1)$.

**Step 7.** The procedure is repeated for the next element of the sequence of step 3 for the recovery of $S(x_2, y_2)$.

**Step 8.** The procedure is repeated for the next groups of step 2, which is not visited earlier.

Recall that $lP$ is represented by $(x_1, y_1)$ and $P_t + l(aP)$ is represented by $S(x_2, y_2)$. In order to pull out $P_t$ from $P_i + l(aP)$, Bob applies his secret key a
and calculates \( a(lP) \) from the first part of the pair, then subtracts it from the second part to obtain: 
\[
P_i + l(aP) - a(lP) = P_i + laP - laP = P_i,
\]
and then reverses the embedding to get back the message.

### 4 Implementation of the proposed Algorithm

The elliptic curve using here is given by the following equation:
\[
y^2 = x^3 - x + 16 \quad [29].
\] (2)

The base point \( P \) is selected as \((5, 7)\). The table below represents a set of all points in the curve:

| 5, 7 | 28, 4 | 18, 1 | 22, 12 |
| 6, 20 | 13, 5 | 2, 14 | 21, 11 |
| 23, 3 | 10, 7 | 14, 22 | 16, 23 |
| 7, 27 | 1, 4 | 0, 4 | 0, 25 |
| 1, 25 | 7, 2 | 16, 6 | 14, 7 |
| 10, 22 | 23, 26 | 21, 18 | 2, 15 |
| 13, 24 | 6, 9 | 22, 17 | 18, 28 |
| 28, 25 | 5, 22 | \( \Omega \) |

Table.1. A set of all points on EC.

Here the choosing curve contains 31 points with \( P \) is the point generator. It is the point witch represents the letter 'a', as well as \( 2P \) represents the letter 'b',..., \( 31P \) represents space.

In our case we use the letters 'a' to 'z' with some of the other symbols like ';', ',', '.', '?' and space for illustration purpose only.

Suppose that Alice wants to encrypt and transmit a message "hello" to Bob, she does the following:

**Step 1. Generating of data sequence**

- \( P \) is a point generator with order \( n=31 \). Then \( m=4 \).
- Convert a sequence 0 to \( n \) to the form:
  
  0000  
  0001  
  0002  
  0011  
  \vdots  
  1011  

- Represent the above form in \((32*4)\) matrix:
F. Amounas and E. H. El Kinani

- Circularly shifting each row of $M$ by one element to the right:

$M = 
\begin{bmatrix}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 2 & 0 \\
0 & 0 & 2 & 1 \\
0 & 0 & 2 & 2 \\
0 & 1 & 0 & 0 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 1 & 1 & 1 \\
0 & 1 & 2 & 0 \\
0 & 1 & 2 & 1 \\
0 & 2 & 0 & 0 \\
0 & 2 & 0 & 1 \\
0 & 2 & 1 & 1 \\
0 & 2 & 2 & 0 \\
0 & 2 & 2 & 2 \\
1 & 0 & 0 & 0 \\
1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 \\
\end{bmatrix}

$M = 
\begin{bmatrix}
0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 \\
2 & 0 & 0 & 1 \\
2 & 0 & 0 & 2 \\
2 & 0 & 0 & 2 \\
2 & 0 & 1 & 0 \\
2 & 0 & 1 & 1 \\
2 & 0 & 1 & 1 \\
2 & 0 & 2 & 0 \\
2 & 0 & 2 & 2 \\
2 & 0 & 2 & 2 \\
1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 \\
\end{bmatrix}$
- The sequence formed is:
  [0000], [1000], [2000], [0001], [1001], [2001], [0002], [1002], [2002], [0010], [1010],
  [2010], [0011], [1011], [2011], [0012], [1012], [2012], [0020], [1020], [2020], [0021],
  [1021], [2021], [0022], [1022], [2022], [0100], [1100], [2100], [0101], [1101]

**Step 2.** Crypting / Decrypting

Hence we shall assume that l = 13, and a = 24. Plaintext is 'h', Therefore:

\[ P_B = aP = 24(5, 7) = (2, 15) \]

\[ P_i = (21, 11) \]

\[ lP_B = 13(2, 15) = (28, 4) \]

\[ P_i + lP_B = (10, 7) \]

\[ IP = 13(5, 7) = (7, 27). \]

Encrypted version of the message is: \( C_m = (S(7, 27), S(10, 7)) \), where \( x_1 = 7, \)
\( y_1 = 27, x_2 = 10, \) and \( y_2 = 7. \)

Apply algorithm (1) for generating the sequence S:

\[ S(7, 27) = 0011 \text{ and } S(10, 7) = 0010 \]

Hence, the message is transformed into the message: 00110010

Then, the transmitted message is: 0000010100000100

The recovery of cipher text is done as follows:

- extract four digits on the cipher text: 0011
- Circularly shifting this sequence by one digit to the left: 0110
- Convert a sequence to decimal form, and store a value in k
  i.e. 0110 will form: \( 0 \times 3^3 + 1 \times 3^2 + 1 \times 3^1 + 0 \times 3^0 = 12 \implies k=12 \)
- Obtain \((k+1)P\) from pre-computed and stored point \((k+1)P=(x_1, y_1)\).
  Therefore \((x_1, y_1) = (7, 27)\) represent \(kP\).

Similarly, other point is recovered by applying a data sequence algorithm1.

Thus we are able to recover the encrypted version: \(((7, 27), (10, 7))\)

From this \(P_i\) should be retrieved using Bobs private key:

\[ 24 (7, 27) = (28, 4) \]

\[ P_i = (10, 7) - (28, 4) = (21, 11) \]

Now reverses the embedding to get back the message. Thus we retrieve the character 'h'.

The table below shows the results for the message "hello". Encryption with data sequence incorporated is shown in Table.2 and decryption is shown in Table.3.
5046
F. Amounas and E. H. El Kinani

Table 2. Encryption before and after applying data sequence

<table>
<thead>
<tr>
<th>Character</th>
<th>Point $P_i$</th>
<th>Encryption before data sequence applied $C_m = (lP, P_i + lP_B)$</th>
<th>Encryption After data sequence applied $C_m = (S(x_1, y_1), S(x_2, y_2))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>h</td>
<td>(21, 11)</td>
<td>((7, 27), (10, 7))</td>
<td>00110010</td>
</tr>
<tr>
<td>e</td>
<td>(6, 20)</td>
<td>((7, 27), (2, 14))</td>
<td>00110002</td>
</tr>
<tr>
<td>l</td>
<td>(16, 23)</td>
<td>((7, 27), (1, 4))</td>
<td>00111011</td>
</tr>
<tr>
<td>l</td>
<td>(16, 23)</td>
<td>((7, 27), (1, 4))</td>
<td>00111011</td>
</tr>
<tr>
<td>o</td>
<td>(0, 4)</td>
<td>((7, 27), (1, 25))</td>
<td>00111012</td>
</tr>
</tbody>
</table>

Hence the message "hello" is transformed into the message:
001100100011000200111011001100111012
So, Alice sends to Bob a serie of bits as following:
00000101000001000000010100000101000001010000010100000101000010100001010000110

When Bob received the above series of bits, he transforms it into serie of digits: 00 ⇒ 0, 01⇒ 1, 10⇒ 2. Then, Bob separates the serie of digits in groups of eight digits and reverses the procedure to recover the coordinates of points.

Decryption of each group is shown in Table 3:

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Reversal of sequence</th>
<th>Decryption</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>00110010</td>
<td>((7, 27), (10, 7))</td>
<td>(21, 11)</td>
<td>h</td>
</tr>
<tr>
<td>00110002</td>
<td>((7, 27), (2, 14))</td>
<td>(6, 20)</td>
<td>e</td>
</tr>
<tr>
<td>00111011</td>
<td>((7, 27), (1, 4))</td>
<td>(16, 23)</td>
<td>l</td>
</tr>
<tr>
<td>00111011</td>
<td>((7, 27), (1, 4))</td>
<td>(16, 23)</td>
<td>l</td>
</tr>
<tr>
<td>00111012</td>
<td>((7, 27), (1, 25))</td>
<td>(0, 4)</td>
<td>o</td>
</tr>
</tbody>
</table>

Table 3. Decryption with reversal of sequence

After decrypting the received message, we obtain the plain text "hello".

5 Conclusion

In this work based on the idea of the ECC, we provide a new method to secure the output of ECC. More precisely, we establish an algorithm to generate a data sequence and use it to the output of our cryptosystem.

Here we apply a data sequence generated by the proposed algorithm to the output of ECC, witch is new in our knowledge. Finally, we like to point out that the use of data sequence will provide better performance in this regard.
ECC encryption and decryption with a data sequence

References


Received: January, 2012