International Journal of Computer Applications Technology and Research
Volume 3– Issue 5, 312 - 317, 2014, ISSN: 2319–8656

Implementation of Matrix based Mapping Method Using Elliptic Curve Cryptography

Geetha G
Dept. of Electronics and Communication
BNM Institute of Technology
Bangalore, India

Padmaja Jain
Dept. of Electronics and Communication
BNM Institute of Technology
Bangalore, India

Abstract: Elliptic Curve Cryptography (ECC) gained a lot of attention in industry. The key attraction of ECC over RSA is that it offers equal security even for smaller bit size, thus reducing the processing complexity. ECC Encryption and Decryption methods can only perform encrypt and decrypt operations on the curve but not on the message. This paper presents a fast mapping method based on matrix approach for ECC, which offers high security for the encrypted message. First, the alphabetic message is mapped on to the points on an elliptic curve. Later encode those points using Elgamal encryption method with the use of a non-singular matrix. And the encoded message can be decrypted by Elgamal decryption technique and to get back the original message, the matrix obtained from decoding is multiplied with the inverse of non-singular matrix. The coding is done using Verilog. The design is simulated and synthesized using FPGA.

Keywords: Cryptography; Elliptic Curve; Finite Field; Mapping; Non-singular matrix; Elgamal Encryption; Elgamal Decryption

1. INTRODUCTION

With the rapid development of technology, people find various methods to hack information. For secured data communication, Cryptography is one of the techniques. It basically deals with encryption and decryption of a given data.

The two types of cryptography being Public and Private key cryptography, where in two types of keys are used in former and a single key is used in later case. The advantage of public key cryptography is that it is more secure than private key cryptography. ECC is one such method of public key cryptography along with RSA. The key attraction of ECC over RSA is that it offers equal security even for smaller bit size, thus reducing the band width, processing complexity [1]. In ECC, the operations such as point inverse, point addition, point subtraction, scalar multiplication are performed on the points obtained from an elliptic curve. These point operations are useful in performing encryption and decryption operations.

In paper [2], Static (One to One) and dynamic (One to N) mapping methods are explained. In static, though it is a simple technique, the same alphanumeric characters from the different words are always mapped onto the same x-y coordinates of the elliptic curve points. When encrypted, points obtained will also be same. So, an intruder can easily interpret data with trial and error method. Hence the secrecy of data transmission by using this methodology is very low. In dynamic mapping, the alphanumeric characters are mapped dynamically on to the points of EC. Thus it is difficult for an intruder to guess which particular character is mapped to which point on EC. But mapping method using matrix method as in paper [3], guarantees the security for the data. And no intruder can hack it. Since this method avoids the regularity in the resultant encrypted text. Thus strengthens the crypto systems and provides better performance.

This paper is organized as follows. The brief introduction to cryptography is given in section 1, cryptography using elliptic curves followed by the point operations, encryption and decryption operations is given in section 2, section 3 describes the proposed method, and the mapping technique followed by illustration and results in section 4, section 5 is about the future enhancements, section 6 gives conclusion and section 7 is about the acknowledgement followed by references.

2. CRYTOGRAPHY USING ELLIPTIC CURVES

2.1. Elliptic Curve

In elliptic curve cryptography, a restricted form of elliptic curve defined over a finite field F_p is considered. One particular interest for cryptography is referred to elliptic group mod p, where p is prime number. Eq.1 defines the condition for choosing the elliptic curve.

$$4a^3+27b^2 \pmod p) \neq 0 \qquad (1)$$

Where ‘a’ and ‘b’ are two nonnegative integers less than p. Then $E_p(a, b)$ indicates the elliptic group mod p whose elements (x, y) are pairs of nonnegative integers less than p. Eq. 2 refers to the general form of elliptic curve.

$$y^2=x^3+ax+b \qquad (2)$$

2.2. Modular Arithmetic

Modular arithmetic is the principal mathematical concept in Cryptography. Here for every operation, modulus is taken w.r.t the prime number. Eg: Prime number considered in this work is 31.

2.3. ECC Point Operations

2.3.1. Point Inverse

If $J = (x, y) \in E(F_p)$, then $(x, y) + (x, - y) = \infty$. The point $(x, - y) \in E(F_p)$ and is called the inverse of J.

Given a point $J(x_1, y_1)$ on an elliptic curve, $-J(x_1, y_1)$ represents its inverse. The inverse of a given point can be computed using Eq. 3.

$$-J(x_1, y_1) = J(x_1, p- y_1) \qquad (3)$$

Fig.1 shows the graphical representation of point inverse.

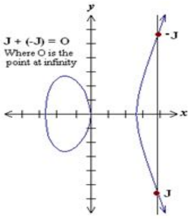


Fig.1. Point Inverse operation on elliptic curve

2.3.2. Point Addition

The Addition operator is defined over E (F_p) and it can be seen that E (F_p) forms an abelian group under addition.

The addition operation in E (F_p) is given by Eq.4.

$$J + \infty = \infty + J = J, \forall J \in E(F_p) \qquad (4)$$

If $J = (x_1, y_1) \in E(F_p)$ and $K = (x_2, y_2) \in E(F_p)$ and $J \neq K$, then $L = J + K = (x_3, y_3) \in E(F_p)$.

Given two points on an elliptic curve, $J(x_1, y_1)$ and $K(x_2, y_2)$, then the addition of those points results in $L(x_3, y_3)$ which lies on the same curve. The graphical representation of point addition is shown in Fig.2. It is computed using Eq. 5, Eq. 6 and Eq. 7 as given in [4] and [5].

$$\lambda = (y_2-y_1)/(x_2-x_1) \qquad (5)$$

$$x_3 = \lambda^2-x_1-x \qquad (6)$$

$$y_3 = \lambda(x_1-x_3)-y_1 \qquad (7)$$

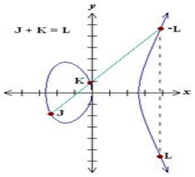


Fig.2. Point addition operation on elliptic curve

2.3.3. Point Doubling

If $J = (x_1, y_1) \in E(F_p)$, then $L = 2J = (x_3, y_3) \in E(F_p)$. Let $J(x_1, y_1)$ be a point on the elliptic curve, then point doubling yields $L(x_3, y_3)$ which lies on that curve. The graphical representation of point doubling is shown in Fig. 3. It is computed using Eq.8, Eq.9 and Eq.10 as given in [4] and [5].

$$\lambda = (3x_1^2 + a) / (2y_1) \qquad (8)$$

$$x_3 = \lambda^2-2x_1 \qquad (9)$$

$$y_3 = \lambda(x_1-x_3)-y_1 \qquad (10)$$

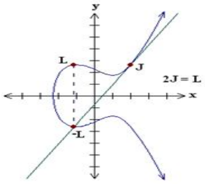


Fig.3. Point doubling operation on elliptic curve

2.3.4. Scalar Multiplication

Given a point $P(x_1, y_1)$ on the curve, to find $k^* P(x_1, y_1)$, where k is any integer, it needs repeated computations of point additions and point doublings.

The reason for choosing prime fields is that distinct additive and multiplicative inverses exist for each number i.e. 0 to (P-1) in the field of the prime number P.

2.4. ECC encryption and decryption

Let E be an elliptic curve defined over a finite field F_p. Now map the plain text on to the points Pm on an elliptic curve. Then the matrix mapping is used for higher security. Later, these points are encrypted which again represents the points on the curve. Then decryption operation is performed.

Elgamal method of encryption consists of following steps:

Step 1: Receiver selects a random integer k, and computes the point kP (‘k’ remains secret).

Step 2: Sender selects a random integer l, and sends the pair of points, (lP, Q+l (kP)) to receiver, here P refers to the generator point.

Step 3: To decrypt the message, receiver finds k(lP) from the first part of the pair, later subtracts it from the second part to get, Q + l(kP) - k(lP) = Q + lkP - klP = Q.

Step 4: Reverse the mapping to get back the original data sent in terms of level I mapped points.

3. PROPOSED METHOD

3.1 To obtain points on an elliptic curve

The elliptic curve $y^2=(x^3+x+13) \bmod 31$ is employed in this work. i.e. by choosing a=1, b=13 and p=31in the general form of elliptic curve given in Eg.2.

The following steps are used to find out the points on an elliptic curve

Step1: Compute y^2 mod 31 for y= 0 to 31.

Step 2: For x= 0 to 31, compute $y^2 =(x^3+x+13) \bmod 31$.

Step 3: Match the value of y^2 in step 2 with that in step 1.

Step 4: If match is found, then the corresponding x and y becomes a point on an elliptic curve.

Step 5: For any point on an elliptic curve, its inverse will also be present.

For the above curve choosen, 34 points can be obtained including point at ∞. Here, the group is said to be cyclic, since the points repeat after 34 points.

The Table 1 gives the set of points on an elliptic curve. Let P be the generator point of the group. Now, the preliminary mapping is performed. I.e. the alphabet in the given message is mapped initially on to the points on an elliptic curve. Thus the alphabet ‘a’ can be mapped as P = (9, 10), ‘b’ can be mapped as 2P = (18, 29), ‘c’ can be mapped as 3P = (23, 19), and so on. Finally the alphabet ‘z’ can be mapped as 26P = (24, 2). Remaining 8 points can be used for mapping special characters or numbers.

Table 1: A set of points on EC

(9,10)	(18,29)	(23,19)	(4,22)	(25,16)
(17,18)	(6,24)	(24,29)	(16,8)	(20,2)
(22,22)	(28,13)	(27,10)	(26,21)	(5,9)
(19,3)	(10,0)	(19,28)	(5,22)	(26,10)
(27,21)	(28,18)	(22,9)	(20,29)	(16,23)
(24,2)	(6,7)	(17,13)	(25,15)	(4,9)
(23,12)	(18,2)	(9,21)	∞	

3.2. Matrix mapping methodology

In this section, a mapping method based on matrices is discussed. The alphabetic characters are mapped on to the points on an elliptic curve. Here, both the sender and receiver agree upon few common relationships among them.

Some of the parameters are defined as follows:-

E (F_p): The set of points on an elliptic curve.

P: Generator point of the curve with order N.

S: Set of the mapping points generated by the proposed algorithm.

A: Non singular matrix, i.e. |A| ± 1 which has only integer entries.
A^{-1}: Inverse of matrix A.
l: Senders private key.
k: Receivers private key.

The following steps are given for matrix mapping method:-
Step 1: Transform the alphabetic characters into points on elliptic curve.
$[P_1(x_1,y_1), P_2(x_2,y_2),\ldots\ldots, P_n(x_n,y_n)]$
Let m be the original message of length n. If n is divided by 3, then the points have to be padded with ∞, which represents space.
Step 2: Create the matrix of 3*r with entries as points on EC. Here, take r =n/3 and s = 2n/3. The matrix M is given as

$$\begin{bmatrix} P1 & P2 & P3 & \ldots & Pr \\ Pr+1 & Pr+2 & Pr+3 & \ldots & Ps \\ Ps+1 & Ps+2 & Ps+3 & \ldots & Pn \end{bmatrix}$$

Step 3: A non singular matrix of 3*3 such that |A| ±1 is selected. Using addition and doubling of points, find Q = AM.
Where, matrix A is given as

$$\begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix}$$

Step 4: The result is set of points S.
$S = [Q_1(x_1,y_1), Q_2(x_2,y_2),\ldots\ldots, Q_n(x_n,y_n)]$

4. ILLUSTRATION AND RESULTS

Choosing non-singular matrix A as

$$\begin{bmatrix} -1 & 5 & -1 \\ -2 & 11 & 7 \\ 1 & -5 & 2 \end{bmatrix}$$

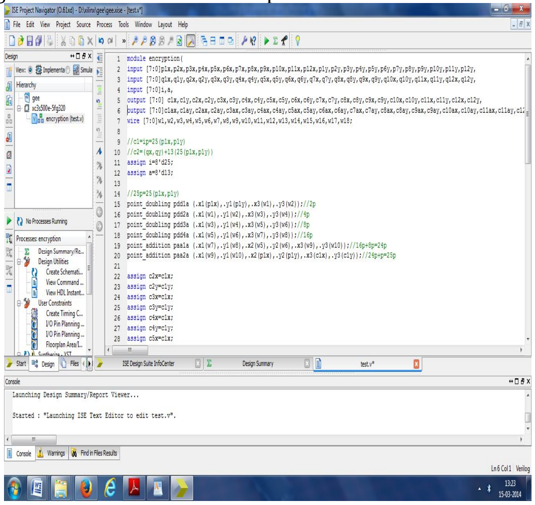
Then, the inverse matrix of A is given by

$$\begin{bmatrix} -57 & 5 & -46 \\ -11 & 1 & -9 \\ 1 & 0 & 1 \end{bmatrix}$$

Let sender's private key l be 25 and receiver's private key k be 13. Now Q = AM yields matrix mapping points, Encrypted points as (C_1, C2) = (lP, Q+l(kP)), Decrypted points D as (C_2-kC_1). The original message can be obtained from decrypted points (D) using the formula M = A^{-1}D.

4.1. Simulation results using Xilinx

The coding is done in Verilog with Xilinx ISE 13.2 simulator. Fig.4 shows the simulation set up.



```
module encryption(
input [7:0]p1x,p2x,p3x,p4x,p5x,p6x,p7x,p8x,p9x,p10x,p11x,p12x,p1y,p2y,p3y,p4y,p5y,p6y,p7y,p8y,p9y,p10y,p11y,p12y,
input [7:0]q1x,q1y,q2x,q2y,q3x,q3y,q4x,q4y,q5x,q5y,q6x,q6y,q7x,q7y,q8x,q8y,q9x,q9y,q10x,q10y,q11x,q11y,q12x,q12y,
input [7:0]i,a,
output [7:0] c1x,c1y,c2x,c2y,c3x,c3y,c4x,c4y,c5x,c5y,c6x,c6y,c7x,c7y,c8x,c8y,c9x,c9y,c10x,c10y,c11x,c11y,c12x,c12y,
output [7:0]c1ax,c1ay,c2ax,c2ay,c3ax,c3ay,c4ax,c4ay,c5ax,c5ay,c6ax,c6ay,c7ax,c7ay,c8ax,c8ay,c9ax,c9ay,c10ax,c10ay,c11ax,c11ay,c1
wire [7:0]w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15,w16,w17,w18;

//c1=ip=25(p1x,p1y)
//c2=(qx,qy)+13(25(p1x,p1y))
assign i=8'd25;
assign a=8'd13;

//25p=25(p1x,p1y)
point_doubling pdd1a (.x1(p1x),.y1(p1y),.x3(w1),.y3(w2));//2p
point_doubling pdd2a (.x1(w1),.y1(w2),.x3(w3),.y3(w4));//4p
point_doubling pdd3a (.x1(w3),.y1(w4),.x3(w5),.y3(w6));//8p
point_doubling pdd4a (.x1(w5),.y1(w6),.x3(w7),.y3(w8));//16p
point_addition paa1a (.x1(w7),.y1(w8),.x2(w5),.y2(w6),.x3(w9),.y3(w10));//16p+8p=24p
point_addition paa2a (.x1(w9),.y1(w10),.x2(p1x),.y2(p1y),.x3(c1x),.y3(c1y));//24p+p=25p

assign c2x=c1x;
assign c2y=c1y;
assign c3x=c1x;
assign c3y=c1y;
assign c4x=c1x;
assign c4y=c1y;
assign c5x=c1x;
```

```
Launching Design Summary/Report Viewer...

Started : "Launching ISE Text Editor to edit test.v".
```

Fig.4. Simulation set up

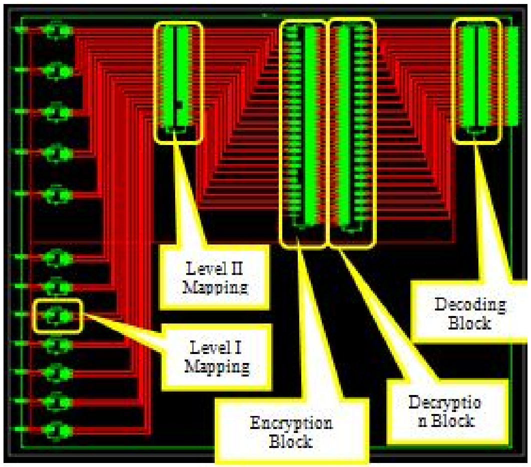


Fig.5. RTL schematic of ECC Top module

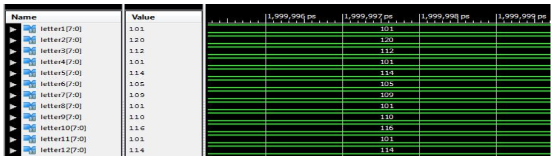
The Fig. 5 gives the RTL schematic of the Top module consisting of level I, level II mappings, encryption, decryption along with decoding.

4.1.1. Addressing letters by its ASCII values

The letters in the given word are addressed by its ASCII values. For the example word "experimenter", level I mapping block is given by Fig.6 with its ASCII values shown in Fig.7 and Fig.8 shows the level I mapping is as discussed in Table 1.

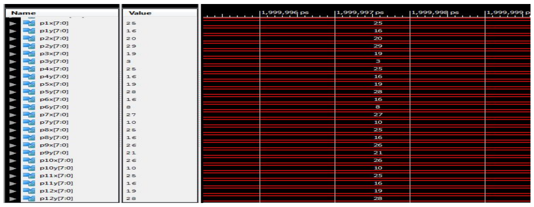
Letter
Level I mapping
Pix
Piy

Fig.6. Level I mapping block.



```
Name             Value
letter1[7:0]     101
letter2[7:0]     120
letter3[7:0]     112
letter4[7:0]     101
letter5[7:0]     114
letter6[7:0]     105
letter7[7:0]     109
letter8[7:0]     101
letter9[7:0]     110
letter10[7:0]    116
letter11[7:0]    101
letter12[7:0]    114
```

Fig.7. Showing the ASCII values for the word "experimenter"



```
Name             Value
p1x[7:0]         25
p1y[7:0]         16
p2x[7:0]         20
p2y[7:0]         29
p3x[7:0]         19
p3y[7:0]         3
p4x[7:0]         25
p4y[7:0]         16
p5x[7:0]         19
p5y[7:0]         28
p6x[7:0]         16
p6y[7:0]         8
p7x[7:0]         27
p7y[7:0]         10
p8x[7:0]         25
p8y[7:0]         16
p9x[7:0]         26
p9y[7:0]         21
p10x[7:0]        26
p10y[7:0]        10
p11x[7:0]        25
p11y[7:0]        16
p12x[7:0]        19
p12y[7:0]        28
```

Fig.8. Respective points for the letters in the chosen word on an elliptic curve

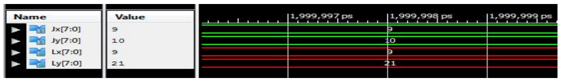
4.1.2. Basic point operations

4.1.2.1. Point Inverse

The point inverse of a point say J = (9, 10) is given by L = (9, 31-10) = (9, 21). Here Fig.9 shows the RTL schematic of Point Inverse and Fig.10 gives its simulation waveform.

Jx
Jy
Point Inverse
Lx
Ly

Fig.9. Block diagram of Point Inverse operation



```
Name      Value   |1,999,997 ps   |1,999,998 ps   |1,999,999 ps
Jx[7:0]   9                        9
Jy[7:0]   10                       10
Lx[7:0]   9                        9
Ly[7:0]   21                       21
```

Fig.10. Point Inverse operation on elliptic curve

4.1.2.2. Point Addition

The point addition of two points say J= (16, 8) and K= (19, 28) yields $x_3 = 6$ and $y_3 = 7$. Fig.11 shows the RTL schematic of point addition and Fig.12 gives its simulation waveform

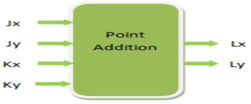
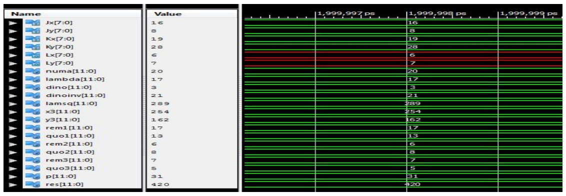


Fig.11. Block diagram of Point addition operation



```
Name              Value   |1,999,997 ps   |1,999,998 ps   |1,999,999 ps
Jx[7:0]           16                       16
Jy[7:0]           8                        8
Kx[7:0]           19                       19
Ky[7:0]           28                       28
Lx[7:0]           6                        6
Ly[7:0]           7                        7
numa[11:0]        20                       20
lambda[11:0]      17                       17
dino[11:0]        3                        3
dinoinv[11:0]     21                       21
lamsq[11:0]       289                      289
x3[11:0]          254                      254
y3[11:0]          162                      162
rem1[11:0]        17                       17
quo1[11:0]        13                       13
rem2[11:0]        6                        6
quo2[11:0]        8                        8
rem3[11:0]        7                        7
quo3[11:0]        5                        5
p[11:0]           31                       31
res[11:0]         420                      420
```

Fig.12. Point addition operation on elliptic curve

4.1.2.3. Point Doubling

When a point is doubled say J = (18, 29) yields $x_3 = 4$ and $y_3 = 22$. Fig.13 shows the RTL schematic of point doubling and Fig.14 gives its simulation waveform.

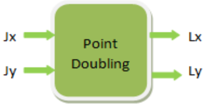
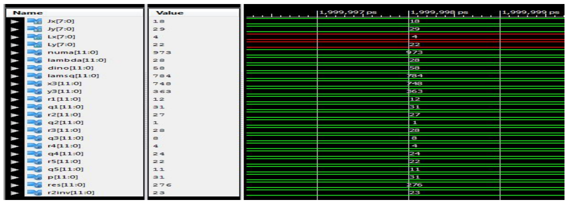


Fig.13. Block diagram of point doubling operation



```
Name              Value   |1,999,997 ps   |1,999,998 ps   |1,999,999 ps
Jx[7:0]           18                       18
Jy[7:0]           29                       29
Lx[7:0]           4                        4
Ly[7:0]           22                       22
numa[11:0]        973                      973
lambda[11:0]      28                       28
dino[11:0]        58                       58
lamsq[11:0]       784                      784
x3[11:0]          748                      748
y3[11:0]          363                      363
r1[11:0]          12                       12
q1[11:0]          31                       31
r2[11:0]          27                       27
q2[11:0]          1                        1
r3[11:0]          28                       28
q3[11:0]          8                        8
r4[11:0]          4                        4
q4[11:0]          24                       24
r5[11:0]          22                       22
q5[11:0]          11                       11
p[11:0]           31                       31
res[11:0]         276                      276
r2inv[11:0]       23                       23
```

Fig.14. Point doubling operation on elliptic curve

4.1.3. Matrix mapping (level II mapping)

After preliminary mapping, the points are again mapped using matrix based mapping approach for high security. Fig.15 refers to the level II mapping block and Fig.16 refers to the matrix mapped points.

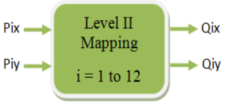
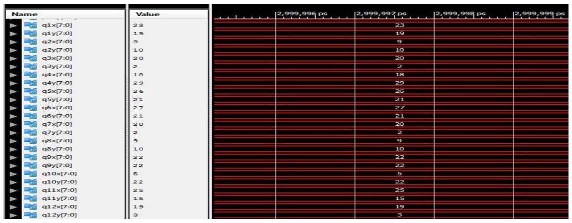


Fig.15. Level II (Matrix) mapping block



```
Name         Value   |2,999,996 ps   |2,999,997 ps   |2,999,998 ps   |2,999,999 ps
q1x[7:0]     23                       23
q1y[7:0]     19                       19
q2x[7:0]     9                        9
q2y[7:0]     10                       10
q3x[7:0]     20                       20
q3y[7:0]     2                        2
q4x[7:0]     18                       18
q4y[7:0]     29                       29
q5x[7:0]     26                       26
q5y[7:0]     21                       21
q6x[7:0]     27                       27
q6y[7:0]     21                       21
q7x[7:0]     20                       20
q7y[7:0]     2                        2
q8x[7:0]     9                        9
q8y[7:0]     10                       10
q9x[7:0]     22                       22
q9y[7:0]     22                       22
q10x[7:0]    5                        5
q10y[7:0]    22                       22
q11x[7:0]    25                       25
q11y[7:0]    15                       15
q12x[7:0]    19                       19
q12y[7:0]    3                        3
```

Fig.16. Matrix mapped points

4.1.4. ECC Encryption

The matrix mapped points are encrypted using the encryption formula given in section II. The Fig.17 refers to ECC encryption block and Fig.18 and Fig.19 refers to its simulation waveform of encrypted points for the example word experimenter.

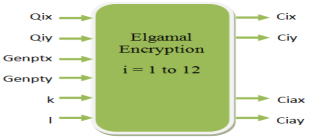
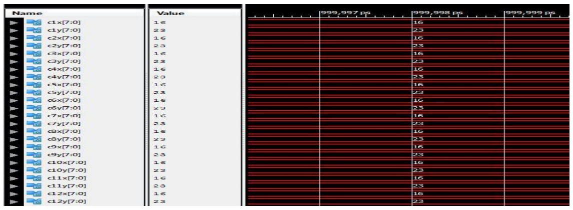
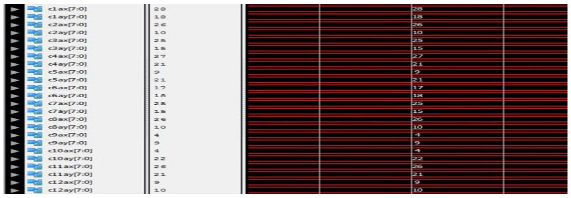


Fig.17. ECC encryption block



```
Name         Value   |999,997 ps   |999,998 ps   |999,999 ps
c1x[7:0]     16                     16
c1y[7:0]     23                     23
c2x[7:0]     16                     16
c2y[7:0]     23                     23
c3x[7:0]     16                     16
c3y[7:0]     23                     23
c4x[7:0]     16                     16
c4y[7:0]     23                     23
c5x[7:0]     16                     16
c5y[7:0]     23                     23
c6x[7:0]     16                     16
c6y[7:0]     23                     23
c7x[7:0]     16                     16
c7y[7:0]     23                     23
c8x[7:0]     16                     16
c8y[7:0]     23                     23
c9x[7:0]     16                     16
c9y[7:0]     23                     23
c10x[7:0]    16                     16
c10y[7:0]    23                     23
c11x[7:0]    16                     16
c11y[7:0]    23                     23
c12x[7:0]    16                     16
c12y[7:0]    23                     23
```

Fig.18. Encrypted points



```
c1ax[7:0]     28     28
c1ay[7:0]     18     18
c2ax[7:0]     26     26
c2ay[7:0]     10     10
c3ax[7:0]     25     25
c3ay[7:0]     15     15
c4ax[7:0]     27     27
c4ay[7:0]     21     21
c5ax[7:0]     9      9
c5ay[7:0]     21     21
c6ax[7:0]     17     17
c6ay[7:0]     18     18
c7ax[7:0]     25     25
c7ay[7:0]     15     15
c8ax[7:0]     26     26
c8ay[7:0]     10     10
c9ax[7:0]     4      4
c9ay[7:0]     9      9
c10ax[7:0]    4      4
c10ay[7:0]    22     22
c11ax[7:0]    26     26
c11ay[7:0]    21     21
c12ax[7:0]    9      9
c12ay[7:0]    10     10
```

Fig.19. Encrypted points

4.1.5. ECC Decryption

The encrypted points are decrypted using the decryption formula discussed in section II. Fig.20 shows the decryption block and Fig.21 refers to the simulation waveform of decrypted points.

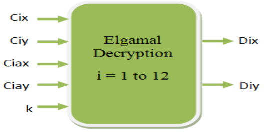
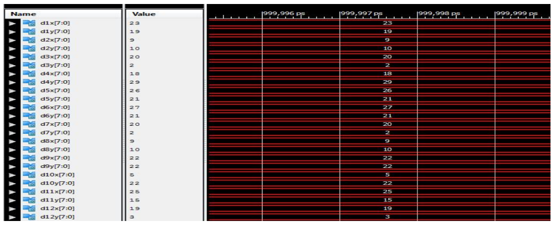


Fig.20. ECC decryption block



```
Name         Value   |999,996 ps   |999,997 ps   |999,998 ps   |999,999 ps
d1x[7:0]     23                     23
d1y[7:0]     19                     19
d2x[7:0]     9                      9
d2y[7:0]     10                     10
d3x[7:0]     20                     20
d3y[7:0]     2                      2
d4x[7:0]     18                     18
d4y[7:0]     29                     29
d5x[7:0]     26                     26
d5y[7:0]     21                     21
d6x[7:0]     27                     27
d6y[7:0]     21                     21
d7x[7:0]     20                     20
d7y[7:0]     2                      2
d8x[7:0]     9                      9
d8y[7:0]     10                     10
d9x[7:0]     22                     22
d9y[7:0]     22                     22
d10x[7:0]    5                      5
d10y[7:0]    22                     22
d11x[7:0]    25                     25
d11y[7:0]    15                     15
d12x[7:0]    19                     19
d12y[7:0]    3                      3
```

Fig.21. Decrypted points

4.1.6. Decoding

After decryption, the original message can be obtained using the formula given in section IV. Fig.22 shows the block diagram of decoding part and Fig. 23 shows the simulation waveform for the same.

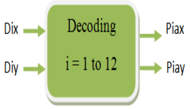
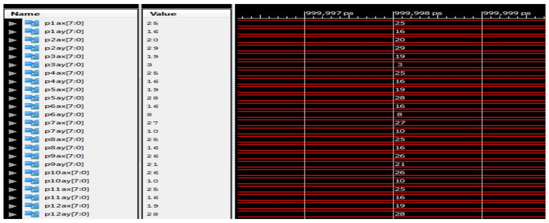


Fig.22. Decoding block



Name	Value
p1ax[7:0]	25
p1ay[7:0]	16
p2ax[7:0]	20
p2ay[7:0]	29
p3ax[7:0]	19
p3ay[7:0]	3
p4ax[7:0]	25
p4ay[7:0]	16
p5ax[7:0]	19
p5ay[7:0]	28
p6ax[7:0]	16
p6ay[7:0]	8
p7ax[7:0]	27
p7ay[7:0]	10
p8ax[7:0]	25
p8ay[7:0]	16
p9ax[7:0]	26
p9ay[7:0]	21
p10ax[7:0]	26
p10ay[7:0]	10
p11ax[7:0]	25
p11ay[7:0]	16
p12ax[7:0]	19
p12ay[7:0]	28

Fig.23. Decoded points

4.2. Results from CADENCE

The design is analysed using Cadence tool. Fig 24 refers to the ECC block, Fig 25 shows level I mapping, Fig 26 gives level II or matrix mapping block, followed by encryption block in Fig 27, decryption block in Fig 28 and Fig 29 refers to the decoding block, point addition block is given by Fig 30 and point doubling block is given by Fig 31.

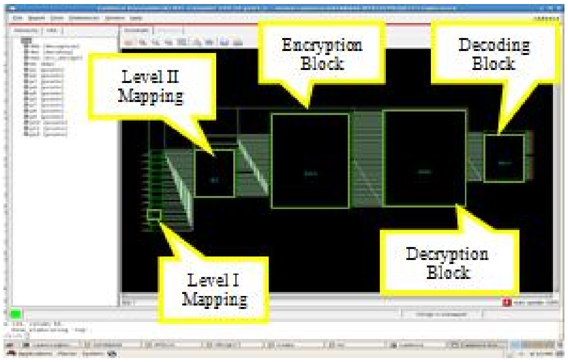


Fig.24. ECC Block using cadence

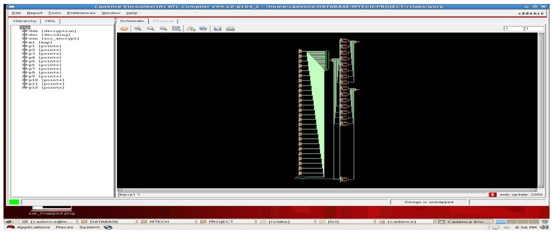


Fig.25. Level I mapping using Cadence

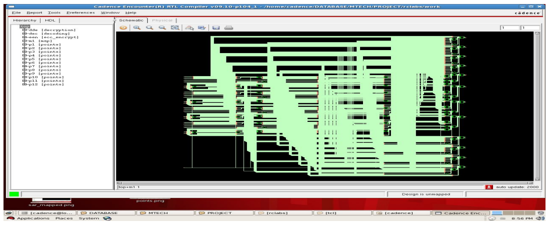


Fig.26. Level II (Matrix) mapping using Cadence

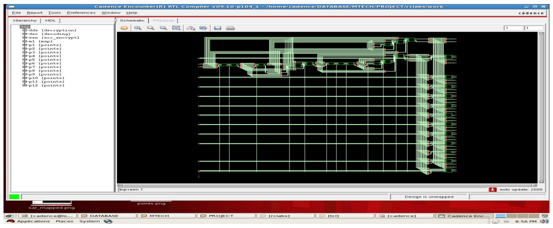


Fig.27.ECC Encryption using Cadence

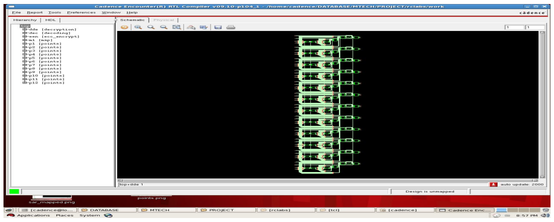


Fig.28.ECC Decryption using Cadence

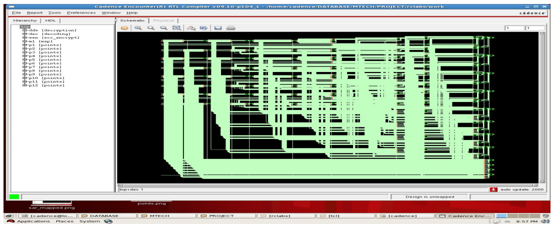


Fig.29.ECC Decoding using Cadence

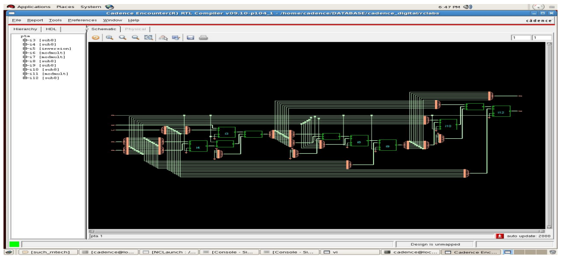


Fig.30.Point addition using Cadence

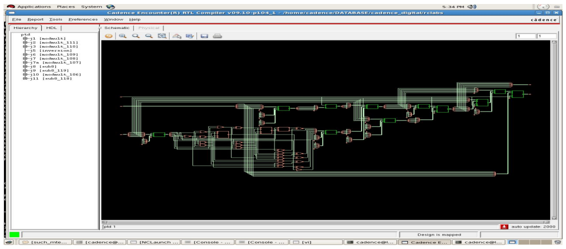


Fig.31.Point doubling using Cadence

4.3. Analysis

In Table 2, the number of point additions, point doublings and point inverses are given for respective blocks.

Table 2: Number of PA, PD and PI's required for each block

Blocks	Point Addition (PA)	Point Doubling (PD)	Point Inverse (PI)
Matrix mapping	40	24	12
Encryption	17	9	0
Decryption	36	36	12
Decoding	60	48	16
TOTAL	**153**	**117**	**40**

The Table 3 gives the area report for point addition and point doubling blocks using Cadence. The Table 4 gives the power report for point addition and point doubling blocks using Cadence. The Table 5 gives the power report for point addition and point doubling blocks using Cadence.

Table 3: Area report using Cadence

Instance	Cells	Cell Area	Net Area	Technology library
Point addition	1597	8755	0	Wireload
Point doubling	1692	8534	0	Wireload

Table 4: Power report using Cadence

Instance	Cells	Leakage power (nW)	Dynamic power (nW)	Technology library
Point addition	1597	27140.155	79593.237	Wireload
Point doubling	1692	25157.323	130700.945	Wireload

Table 5: Timing report using Cadence

Instance	Fan out	Load (fF)	Slew (ps)	Delay (ps)	Arrival(ps)
Point addition	15	18.8	0	+90	20520 R
Point doubling	18	23.5	0	+90	30014 R

5. FUTURE WORK

Optimizing the design in terms of Area, Power and Speed and extending the work so that the numerals, capital letters etc also can be encoded.

6. CONCLUSION

In this work, a method of mapping alphabetic characters to an elliptic curve points by using a non-singular matrix is described. The mapping points are encrypted and decrypted using ECC technique. The obtained results show that the chosen method avoids the regularity in the resultant encrypted points. The Table 6 gives the encrypted and decrypted points for the example word “experimenter”.

Table 6: Encrypted and decrypted points for the word “experimenter”

. Char	Point P_m	Matrix Mapped points Q	Encrypted points (C1,C2)	Decrypted points D
e	(25,16)	(23,19)	((16,23),(28,18))	(23,19)
x	(20,29)	(9,10)	((16,23),(26,10))	(9,10)
p	(19,3)	(20,2)	((16,23),(25,15))	(20,2)
e	(25,16)	(18,29)	((16,23),(27,21))	(18,29)
r	(19,28)	(26,21)	((16,23),(9,21))	(26,21)
i	(16,8)	(27,21)	((16,23),(17,18))	(27,21)
m	(27,10)	(20,2)	((16,23),(25,15))	(20,2)
e	(25,16)	(9,10)	((16,23),(26,10))	(9,10)
n	(26,21)	(22,22)	((16,23),(4,9))	(22,22)
t	(26,10)	(5,22)	((16,23),(4,22))	(5,22)
e	(25,16)	(25,15)	((16,23),(26,21))	(25,15)
r	(19,28)	(19,3)	((16,23),(9,10))	(19,3)

In the paper [2], an intruder can easily guess the repeating letter since the mapping methods discussed shows regularity in the encrypted points. The mapping method employed in this paper does not show any regularity. Hence it would be difficult to guess the word. Thus it is concluded that the proposed mapping method can not only strengthen the crypto system but it also guarantee the confidentiality of messages hence providing better performance in this regard.

7. ACKNOWLEDGMENTS

Our thanks to the BNMIT management who have contributed towards this paper.

8. REFERENCES

[1] A Comparative Study of Public Key Cryptosystem based on ECC and RSA, Arun kumar, Dr. S.S. Tyagi, Manisha Rana, Neha Aggarwal, Pawan Bhadana, Manav Rachna International University, Faridabad, India, International Journal on Computer Science and Engineering (IJCSE), 2011.

[2] Efficient Mapping methods for Elliptic Curve Cryptosystems, O.Srinivasa Rao, Prof. S. Pallam Setty, Andhra Pradesh, India, International Journal of Engineering Science and Technology, 2010.

[3] Fast Mapping Method based on Matrix Approach for Elliptic Curve Cryptography, F. Amounas and E.H. El Kinani, Moulay Ismaïl University, Morocco, International Journal of Information & Network Security (IJINS), Vol.1, No.2, June 2012, pp. 54~59, ISSN: 2089-3299.

[4] William Stallings, “Cryptography and network security principles and practice” , *Prentice Hall,5th Edition, 2011*

[5] Darrel R. Hankerson, A. Menezes and A. Vanstone, “Guide to Elliptic Curve Cryptography”, *Springer, 2004.*

[6] http://en.wikipedia.org/wiki/Elliptic_curve_cryptography

[7] http://www.certicom.com/index.php/ecc-tutorial

[8] http://www.eccworkshop.org/Engineering,UK, 2009