Encoding And Decoding of a Message in the Implementation of Elliptic Curve Cryptography using Koblitz's Method

Padma Bh[1], D.Chandravathi[2] , P.Prapoorna Roja[3]

1.Asst.Professor,Dept.of MCA,GVP College for Degree &PG Courses,Visakhapatnam.
2.Asst.Professor,Dept of MCA, GVP College for Degree & PG Courses, Visakhapatnam.
3.Professor, SSN College Of Engineering, Department Of IT, Chennai.

Abstract:
Elliptic Curve Cryptography recently gained a lot of attention in industry. The principal attraction of ECC compared to RSA is that it offers equal security for a smaller bit size, thereby reducing processing overhead. ECC is ideal for constrained environment such as pager, PDAs, cellular phones and smart cards. For the implementation of elliptic curve cryptography (ECC) the plaintext encoding should be done before encryption and decoding should be done after decryption. ECC Encryption and Decryption methods can only encrypt and decrypt a point on the curve and not messages. The Encoding(converting message to a point) and Decoding (converting a point to a message) are important functions in Encryption and Decryption in ECC. The paper discusses Koblitz's method to represent a message to a point and vice-versa. The paper also describes implementation results of Koblitz's Encoding and Decoding methods.

Keywords: Encryption, Decryption, Elliptic Curve cryptography, Encoding, Decoding.

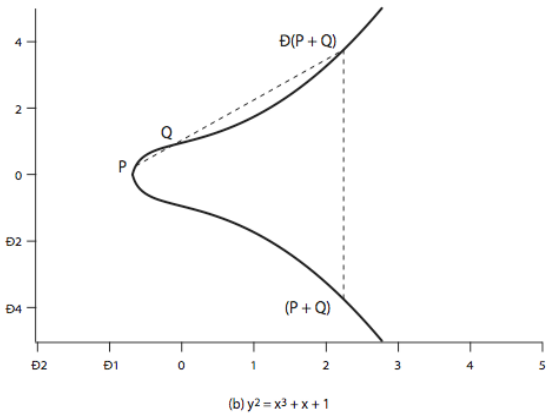
Introduction:
Elliptic Curve Cryptography[2] is a public key Cryptography. ECC is ideal for environments such as pager, PDAs, cellular phones and smart cards. Moreover, because of the apparent hardness of the underlying elliptic curve discrete logarithm problem (ECDLP), ECC systems are also well suited for applications that need long-term security requirements. Elliptic Curve Cryptography (ECC) is a public key technology that offers performance advantages at higher security levels. Every user taking part in public key cryptography will take a pair of keys, a public key and a private key. Only the particular user knows the private key whereas the public keys are distributed to all users taking part in the communication. Some public key algorithm may require a set of predefined constants to be known by all the devices taking part in the communication. In ECC we call these predefined constants as 'Domain Parameters.

Understanding ECC needs full mathematical background on elliptic curves. Elliptic curves are not ellipses. The general cubic equation of elliptic curves is $y^2+axy+by=x^3+cx^2+dx+e$. But for our purpose it is sufficient to limit the equation to the form $y^2 = x^3 + ax + b$. Say $E_P(a,b)$ consisting of all the points (x,y) that satisfy the above equation together with element at infinity O. A group can be defined based on the set $E_P(a,b)$ for specific values of a and b[8]. If P,Q R are points on $E_P(a,b)$ the relations commutativity, associativity, existence of an identity element and existence of inverse hold good[4]. The heart of ECC is discrete logarithm problem that can be stated as "it should be very hard to find a value k such that **Q=kP** where P and Q are known'. But 'it should be relatively easy to find Q where k and P are known' P, Q are points on the elliptic curve [5].

Elliptic Curve Example:

Let the equation of the curve is
$y^2 \mod p = x^3 + ax + b \mod p$
Inputs : a, b, p (p is key of the ECC algorithm)
Choose two non-negative integers a, b and a large prime number such that $4a^3 + 27b^2 \mod p \neq 0$. For Example, the following figure (fig 1) shows the elliptic curve, $y^2 \mod 23 = x^3 + x + 1 \mod 23$. Here points P, Q lie on the curve and P+Q gives another point that lie on the line that connects P and Q as shown in the fig 1 below.

Fig 1



The set of points on the above curve are
{
[0 1],[0 22] ,[1 7], [1 16], [3 10], [3 13],[5 4], [5 19] ,[6 4], [6 19] ,[7 11],[7 12], [9 7],[9 16],
[11 3],[1120],[12 4],[12 19],
[13 7],[13 16],[17 3],[17 20],
[18 3],[18 20],[19 5],[19 18]
}

Multiplication of a point with a positive integer k is defined as the sum of copies of P, k times. This operation is called Point Multiplication[1] in ECC. So 3P=P+P+P.

The above points form the Group ie Ep(a,b). Each X and Y coordinate ranges between 0 and 22 . The addition of the two points on the curve and the inverse of a point on the curve are defined in the field using modular arithmetic. The point at the infinity is identity point on the curve.

ECC Public Key Cryptosystem
In the public key elliptic curve cryptosystems, we assume that entity A wants to send a message m to entity B securely. Order of a point on the curve can be defined as a value n such that
nP = P+P+..+P.. n times = O (infinity)[9].

Key generation:
Both the entities in the cryptosystem agree upon a,b,p,G,n which are called 'Domain Parameters' of ECC[3]. G is called generator point and n is the order of G. Now A generates a random number $n_A < n$ as his private Key and calculates his public key Set $P_A = G+G+G\ldots+n_A$ times. B generates a random number nB < n as his private Key and calculates his public key, set $P_B = G+G+G\ldots+n_B$ times.

Key Exchange:
Entity A computes his Shared Key by Computing $K = P_A + P_A +..+\ldots n_B$ times
Entity A computes his Shared Key by Computing $K = P_B + P_B +..+\ldots n_A$ times
The two above keys have same value because:
$n_A*P_B = n_A* (n_B*G)= n_B*(n_A *G)= n_B * P_A$

Encryption:
A sends C_m = 2 ciphertext points those are { kG, $P_m + k P_B$) }.
Where G - generator Point
P_m - plaintext point on the curve
k - a random number chosen by A
P_B - public key of B

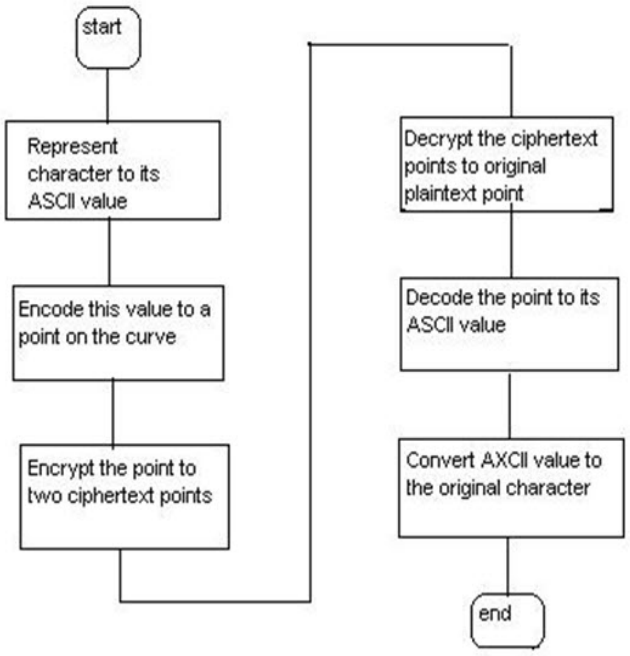
Decryption:
$P_m + kP_B - n_B(kG) = Pm + k(n_B)G - n_B(kG) = P_m$

Encoding and Decoding a message in the implementation of ECC

ECC Encryption and Decryption methods can only encrypt and decrypt a point on the curve not messages. Unfortunately, there are no known polynomial time algorithms for finding a large number of points on an arbitrary curve. We are not simply looking for random points on E, here. We want a systematic way of finding points on Ep(a,b) relating somehow to the plaintext message. Therefore, we are forced to use probabilistic algorithms to do this, where the chance of failure is acceptably small. Thus Encoding(message to a point) and Decoding (point to a message) methods are important while Encryption and Decryption.

Message Encoding and Decoding
Let us suppose a text file has to be encrypted, a user can encrypt the ASCII code of each and every printable character on the keyboard , let us say he has to encrypt an 8-bit number , can represent 128 characters on the keyboard. Fig 2 shows the sequence of steps to be followed when a message to be encrypted and decrypted using elliptic Curve Cryptography.

All the points on the elliptic curve can be directly mapped to an ASCII value, select a curve on which we will get a minimum of 128 points, so that we fix each point on the curve to an ASCII value. For example, 'ENCRYPT' can be written as sequence of ASCII characters that is ' 69' '78' '67' '82' '89' '80' '84' we can map these values to fixed points on the curve. This is easiest method for embedding a message but less efficient in terms of security. The steps to be followed during encoding and decoding are given the following flowchart ie Fig 2
Fig 2.



Koblitz's Method for Encoding Plaintext[6]:

Step1: Pick an elliptic curve $E_p(a,b)$.
Step 2: Let us say that E has N points on it.
Step 3: Let us say that our alphabet consists of the digits 0,1,2,3,4,5,6,7,8,9 and the letters A,B,C,. . . , X,Y,Z coded as 10,11,. . . , 35.
Step 4: This converts our message into a series of numbers between 0 and 35.
Step 5: Now choose an auxiliary base parameter, for example $k = 20$. (both parties should agree upon this)
Step 6: For each number *mk* (say), take $x=mk + 1$ and try to solve for *y*.
Step 7: If you can't do it, then try $x = mk +2$ and then $x = mk +3$ until you can solve for *y*.
Step 8: In practice, you will find such a *y* before you hit $x = mk + k - 1$. Then take the point (x,y). This now converts the number m into a point on the elliptic curve. In this way, the entire message becomes a sequence of points.

Decoding:
Consider each point (x,y) and set *m* to be the greatest integer less than $(x-1)/k$. Then the point (x,y) decodes as the symbol *m*.

Example:
Say the parameters of curve are:
p(751),a(-1),b(188),n(727).

1. Say we have to send character 'b'.
2. 'B' is first encoded as number 11.
3. x=mk+1 ie 11*20+1=221cannot solve it for a y such that $y^2= x^3 + ax+ b \mod p$.
4. So go for x=mk+2 , x=222 , no y exists. x=mk+3, x=223, no y exists.
5. x=mk+4 so x=224 can solve it for y and y=248.
6. Now the point (224,248) is point is encrypted and decrypted as a message.
7. To decode just compute (x-1)/k ie (224-1)/20=223/20 ie 11.15.
8. Return 11 as original plaintext(greatest integer less than (x-1)/k ,that is 11.
9. The number 11 is now decoded to character 'B'.
10. The probability that we fail to find a square (and hence fail to associate m to a point) is about $1/2^k$[10].

How to select Curve Parameters for Koblitz's Method:
In Koblitz's method the maximum possible value for m is 128, if an 8-bit number is encrypted. Say value of k=10. Now the minimum value of x is mk+1 ie 128*10+1=1280 to represent a character.

To get a point on the curve whose x-coordinate is above 1280, we need to select an elliptic curve with p value not less than 1280. So depending on the value of k(>=10) we need to select the curve parameters.

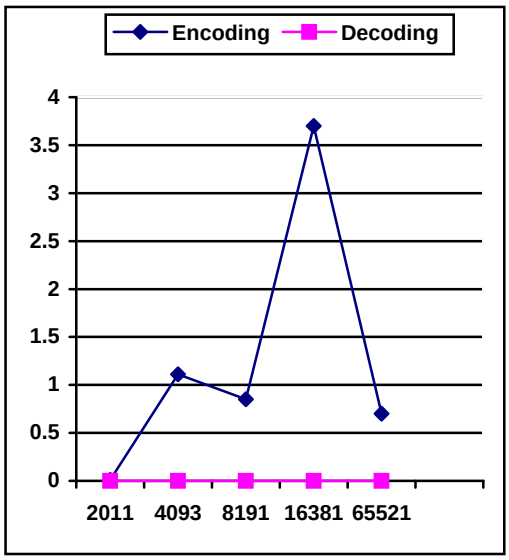
Implementation Results :
The Encoding and Decoding times are specific to the processor. The observations are recorded on a machine with 1GB RAM and 1.6 GHz processor speed on Win XP platform. The following table (table 1) shows CPU Times for Encoding and Decoding when implemented in MATLAB[7], and a character 'a' is being encrypted and decrypted for different domain parameters of elliptic curves.

Table 1.

p	a	b	CPU TIMES(secs)	
			Encoding	Decoding
2011	9	7	0.01	0.000001
4093	9	7	1.11	0.000003
8191	10	17	0.85	0.000003
16381	1	17	3.7	0.000002
65521	7	29	0.7	0.000004

The above CPU times for encoding and decoding are show the following graph fig 3.

Fig 3.



Conclusions:
The Execution time for encoding and decoding functions will not vary according to the value of a,b,p(domain parameters ecc).The execution time for encoding is different for different values of ecc domain parameters. The Execution time taken for decoding is constant for different values of a,b,p. The Execution time for Decoding is negligible compared to that of Encoding.

References:

[1] Darrel Hankerson, Julio Lopez Hernandez, Alfred Menezes, *Software Implementation of Elliptic Curve Cryptography over Binary Fields*, 2000, Available at http://citeseer.ist.psu.edu/hankerson00software.html
[2] Certicom, Standards for Efficient Cryptography, *SEC 1: Elliptic Curve Cryptography, Version 1.0*, September 2000, Available at http://www.secg.org/download/aid-385/sec1_final.pdf
[3] Certicom, Standards for Efficient Cryptography, *SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0*, September 2000, Available at http://www.secg.org/download/aid-386/sec2_final.pdf
[4] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
[5] William Stallings, *Cryptography and Network Security, Principles and Practice.* ed.,Prentice Hall, New Jersey,2003.
[6]. N. Koblitz, \Elliptic curve cryptosystems", *Mathematics of Computation*, 48 (1987), 203-209.
[7] MATLAB Summary and Tutorial, *www.math.ufl.edu/help/matlab-tutorial*
[8] R. Schoof. Elliptic Curves Over Finite Fields and the Computation of Square Roots mod p. Mathematics of Computation, Vol. 44, No. 170, pp. 483-494, April 85.
[9] F. Morain. Building cyclic elliptic curves modulo large primes. Advances in Cryptology - EUROCRYPT '91, Lecture Notes in Computer Science, 547: 328-336,1991.
[10] N. Koblitz. *A Course in Number Theory and Cryptography*, Springer-Verlag, second edition, 1994.