Implementation of ElGamal Elliptic Curve Cryptography over prime field using C

Conference Paper · February 2014

DOI: 10.1109/ICICES.2014.7033751

1 author:

Monjul Saikia
North Eastern Regional Institute of Science and Technology
59 PUBLICATIONS **189** CITATIONS

SEE PROFILE

Implementation of ElGamal Elliptic Curve Cryptography Over Prime Field Using C

Debabrat Boruah
Department of Computer Science & Engineering
NERIST, Nirjuli-791109, Arunachal Pradesh, India
dboruah78@gmail.com

Monjul Saikia
Department of Computer Science & Engineering
NERIST, Nirjuli-791109, Arunachal Pradesh, India
monjuls@gmail.com

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 key size, thereby reducing processing overhead. There is no sub exponential time algorithm in solving the Elliptic curve discrete logarithm problem. ElGamal Elliptic Curve Cryptography is a public key cryptography analogue of the ElGamal encryption schemes which uses Elliptic Curve Discrete Logarithm Problem. The ElGamal Elliptic Curve Cryptosystem is implemented using C language in our work. We divided the whole cryptosystem into seven different phases. The paper also describes different efficient algorithms used in the implementation to perform various mathematical manipulations.**

Keywords– ElGamal; Elliptic Curve Cryptography; Finite field; public key; Encryption; Decryption.

I. INTRODUCTION

Cryptography is the science of protecting information by encrypting them into unreadable format, called cipher text. Only those who possess a secret key can decipher (or decrypt) the message into plaintext. Public-key cryptography and symmetric-key cryptography are two main categories of cryptography. Symmetric Key scheme requires sharing of a secret key between the parties who want to make communications. This secret key is used in both encryption and decryption processes. On the other hand Public Key schemes use different keys for encryption and decryption. Public key cryptography contains two keys, which are public and private keys. Only the particular user knows the private key whereas the public keys are distributed to all users taking part in the communication. The well-known public-key cryptography algorithms are RSA (Rivest, et al. 1978)[3], El-Gamal and Elliptic Curve Cryptography[5].

In 1976, Diffie Hellman introduced the concept of public key cryptosystem by describing a public key distribution scheme. The security of this scheme based on intractability of discrete logarithm problem in the multiplicative group of a large finite field. On the other hand the strength of RSA[3] lies in integer factorization problem. That is when we are given a number; we have to find its prime factors. It becomes quite complicated when dealing with large numbers. This is the strength of RSA and to an extent, is the disadvantage associated with it.

In 1985, ElGamal made use of discrete logarithm problem to construct a practical public key cryptosystem with security equivalent to Diffie Hellman scheme. Although the discrete logarithm problem referred explicitly to the problem of finding logarithm respect to a primitive element in the multiplicative group of the field modulo a prime p, this idea can be extended to other arbitrary groups with the difficulty of the problem varying with representation of the group. So the cryptosystem ElGamal can be generalized to an arbitrary finite cyclic group.

In 1985, Kobiltz[2] and Miller[1] independently proposed the implementation of public key cryptosystem using elliptic curve group over finite field which is known as Elliptic curve cryptography(ECC). This elliptic curve forms finite cyclic group so that the elliptic curves groups over finite field can be used to implement ElGamal public key cryptosystem. ElGamal Elliptic Curve Cryptography is a public key cryptography analogue of the ElGamal encryption schemes which uses Elliptic Curve Discrete Logarithm Problem. Elliptic curves discrete logarithm problem appear to be much harder than the discrete logarithm problem in other groups(DSA) and integer factorization problem(RSA). Hence elliptic curve cryptosystem can match the security of other cryptosystems while using smaller key. For example an elliptic curve cryptosystem with public key of size 160 bits is as secure as RSA and DSA cryptosystems with public key of size 1024 bits. With smaller key size, elliptic curve cryptosystem can offer lower memory requirement, faster implementation and lower bandwidth requirement.

There are lots of research works going on in recent years on the ECC. The ElGamal ECC implementation described in [4] and [5] mainly focuses on the elliptic curve point manipulations for encryption and decryption over finite fields. ECC encryption and decryption process can only encrypt and decrypt a point on the curve and not messages. The encoding (converting message to a point), decoding (converting a point to a message) and public key validation are important functions in encryption and decryption in ECC. In this paper we have described different algorithms and their implementation for the proposed ElGamal Elliptic Curve Cryptosystem in C programming language. The paper discusses Koblitz's method to represent a message to a point and vice versa which is used in our cryptosystem.

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

II. ELLIPTIC CURVE CRYPTOGRAPHY

Elliptic Curve Cryptography (ECC) is a public key cryptography. In public key cryptography each user or the device taking part in the communication generally have a pair of keys, a public key and a private key, and a set of operations associated with the keys to do the cryptographic operations. Only the particular user knows the private key whereas the public key is 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. '*Domain parameters'* in ECC is an example of such constants. Public key cryptography, unlike private key cryptography, does not require any shared secret between the communicating parties but it is much slower than the private key cryptography.

Understanding ECC needs full mathematical background on elliptic curves. 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 $y^2 = x^3 + ax + b$ where $4a^3 + 27b^2 \neq 0$.

Let $E_p(a,b)$ be the set consisting of all the points (x,y) that satisfy the above equation together with element at infinity O. An *abelian group* can be defined based on the set $E_p(a,b)$ over addition operation for specific values of a and b[8]. If P,Q and R are points on $E_p(a,b)$ the relations commutativity, associativity, existence of an identity element and existence of inverse hold good[8]. Such a group can then be used to create an analogue of the discrete logarithm problem which is the basis for ElGamal public key cryptosystems. Each value of the *'a'* and *'b'* gives a different elliptic curve. The public key is a point in the curve and the private key is a random number in the interval *[1, n-1],'n'* is the curve's order. The public key is obtained by multiplying the private key with the generator point G in the curve. The generator point G is the point on the curve. The generator point G, the curve parameters *'a'* and *'b'*, together with few more constants constitutes the domain parameters of ECC.

III. ELLIPTIC CURVE DISCRETE LOGARITHM PROBLEM

The security of ECC depends on the difficulty of Elliptic Curve Discrete Logarithm Problem. Let P and Q be two points on an elliptic curve such that $kP = Q$, where k is a scalar. Given P and Q, it is computationally infeasible to obtain k, if k is sufficiently large. But it is relatively easy to find Q where k and P are known. k is the discrete logarithm of Q to the base P. Thus, point multiplication is the basic operation in ECC. For example, the multiplication of a scalar *'k'* with any point *'P'* on the curve in order to obtain another point *'Q'* on the curve.

IV. FINITE FIELDS

Generally in mathematics elliptic curve operations are defined over real numbers. However, operations over the real numbers are inaccurate and slow due to round off errors, whereas cryptographic operations need to be accurate and fast. To make operations on elliptic curve accurate and more efficient, the curve cryptography is defined over two finite fields: Prime field F_p and Binary field F_2^m

The field is chosen with finitely large number of points suited for cryptographic operations. In our work we have implemented elliptic curve over Prime finite field for the ElGamal Elliptic Curve Cryptosystem. So the following discussions will be bound to the elliptic curves over prime finite field only. The operations in these sections are defined on affine coordinate system [12]. Affine coordinate system is the normal coordinate system that we are familiar with in which each point in the coordinate system is represented by the vector (x, y).

A. Elliptic Curve on Prime field F_p:

The equation of the elliptic curve on a prime field F_p is $y^2 \bmod p = x^3 + ax + b \bmod p$, where $4a^3 + 27b^2 \bmod p \neq 0$. p is a prime number. Here the elements of the finite field are integers between 0 and $p - 1$. All the operations such as addition, subtraction, division, multiplication involves integers between 0 and $p - 1$. This is modular arithmetic. The prime number p is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC[1] specifies curves with p ranging between 112 to 521 bits.

B. Point Multiplication

Scalar point multiplication is a block of all elliptic curve cryptosystems. It is an operation of the form $k.P$. *'P'* is a point on the elliptic curve and *'k'* is a positive integer.

Computing $k.P$ means adding the point *'P'* exactly $k-1$ times to itself, which results in another point *'Q'* on the same elliptic curve. Point multiplication uses two basic elliptic curve operations:

a) Point addition (add two points to find another point)

b) Point doubling (adding point P to itself to find another point)

For example to calculate $kP=Q$ if k is 23 then $kP=23P=2(2(2(2P) + P) + P) + P$ so to get the result, point addition and point doubling is used repeatedly.

The above method of point multiplication which uses point addition and point doubling repeatedly to find the result is known as 'double and add' method. There are other efficient methods for point multiplication such as NAF (Non – Adjacent Form) and Binary NAF method for point multiplication. In our work we have used NAF as well as Binary NAF methods for point multiplication.

C. Point Addition:

Consider two distinct points P and Q such that $P = (x_1, y_1)$ and $Q = (x_2, y_2)$.

Let $R = P + Q$ where $R = (x_3, y_3)$, then

$x_3 = \lambda^2 - x_1 - x_2 \bmod p$

$y_3 = \lambda(x_1 - x_3) - y_1 \bmod p$

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

$\lambda = (y_2 - y_1)/(x_2 - x_1) \bmod p$, λ is the slope of the line through P and Q.

If $Q = -P$ i.e. $Q = (x_1, -y_1 \bmod p)$ then $P + Q = O$. where O is the point at infinity.

If $Q = P$ then $P + Q = 2P$ then point doubling equations are used. Also $P + Q = Q + P$.

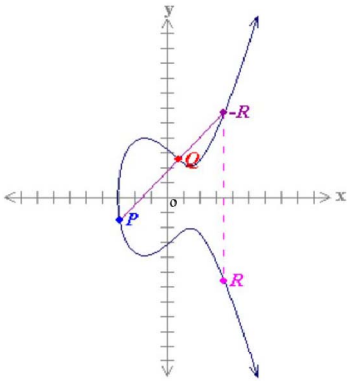

Figure 1: Addition of two points on the elliptic curve.

Consider P and Q be two different points on the elliptic curve E as shown in fig1. Since we are crossing a line with a cubic curve, the line that connects P and Q must intersect through a third point on the curve; the point is noted as -R. Another point called R will be resolved from the reflection of this point -R in the x-axis, where R = P+Q.

D. Point Doubling

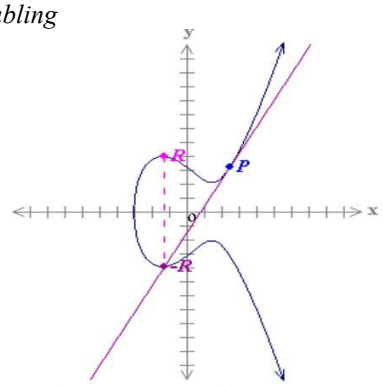

Figure 2: Point Doubling

Consider a point P in the elliptic curve such that

$P = (x_1, y_1)$, where $y_1 \neq 0$.

Let $R = 2P$ where $R = (x_2, y_2)$, then

$x_2 = \lambda^2 - 2x_1 \bmod p$

$y_2 = \lambda(x_1 - x_2) - y_1 \bmod p$

$\lambda = (3x_1^2 + a) / (2y_1) \bmod p$, λ is the tangent at point P and a is one of the parameters chosen with the elliptic curve.

If $y_1 = 0$ then $2P = O$, where O is the point at infinity.

To double a point P on the elliptic curve, a tangent line to the curve and passing by P is taken as shown in fig2. The line must cross the curve through another point; the point is noted as -R. Then we reflect the point –R in the x-axis to the point R where R=2P.

E. Point Subtraction

Consider two distinct points P and Q such that P = (x_1, y_1) and Q = (x_2, y_2). Then P - Q = P + (-Q) where -Q = $(x_2, -y_2 \bmod p)$.Point subtraction is used in certain implementation of point multiplication such as NAF.

V. Modular Arithmetic

Modular arithmetic over a number p involves arithmetic between numbers 0 and $p - 1$. If the number happens to be out of this range in any of the operation the result is wrapped around into the range 0 and $p - 1$.

A. Addition

Let $p = 23, a = 15, b = 20$

$a + b \pmod p = 15 + 20 \pmod{23} = 35 \bmod 23 = 12$

Since the result of $a + b = 35$ which is out of the range $[0,22]$, the result is wrapped around into the range $[0,22]$ by subtracting 35 with 23 till the result is in range $[0,22]$. $a \bmod b$ is thus explained as remainder of division a/b.

B. Subtraction

Let $p = 23, a = 15, b = 20$

$a - b \pmod p = 15 - 20 \pmod{23} = -5 \bmod 23 = 18$

Since the result of $a - b = -5$ which is negative and out of the range $[0,22]$, the result is wrapped around into the range $[0,22]$ by adding -5 with 23 till the result is in range $[0,22]$.

C. Multiplication

Let $p = 23, a = 15, b = 20$

$a * b \pmod p = 15 * 20 \pmod{23} = 300 \bmod 23 = 1$

Since the result of $a * b = 300$ which is out of the range $[0,22]$, The result is wrapped around into the range $[0,22]$ by subtracting 300 with 23 till the result is in range $[0,22]$.

D. Division

The division $a/b \pmod p$ is defined as $a * b^{-1} \pmod p$.b^{-1} is the multiplicative inverse of b over p.

E. Multiplicative Inverse

Multiplicative inverse of number b with respect to $mod\ p$ is defined as a number b^{-1} such that $b*b^{-1} \pmod p = 1$. The algorithm such as Extended Euclidean algorithm[12][8] can be used to find the multiplicative inverse of a number efficiently. Finding multiplicative inverse is a costly operation.

F. Finding x mod y

$x \bmod y$ is the remainder of the division x/y. Finding $x \bmod y$ by repeatedly subtracting y with x till the result is in range $[0,y-1]$ is a costly operation. Methods such as Barrett Reduction[15] can be used to find modulus of a number in efficient manner.

VI. Elliptic Curve Diffie-Hellman (ECDH) Key Exchange

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

ECDH operates by providing the two parties sharing a secret key with a public key, which in this case is a point *P* on elliptic curve *E*. Alice performs scalar multiplication using this point *P* and a random number *a*, which is the secret key of Alice. *a.P* now becomes public key of Alice which she can share with the other party. On the other end, Bob performs scalar multiplication using point *P* and a random number of his choice i.e. *b*, which is secret key of Bob. *b.P* becomes public key of Bob which he shares with Alice. Alice performs scalar multiplication of public key of Bob*(b.P)* with her secret key *a* to get *a.b.P*. Bob also does the same with his secret key *b* and public key of Alice(*a.P)* to get the same *a.b.P*. This entity i.e. *a.b.P* is same for both the parties and is their shared key.

VII. ECC DOMAIN PARAMETERS OVER F_P

As stated earlier the domain parameters are the parameters or constants to be agreed upon or known by all the devices taking part in cryptosystem [7]. Elliptic curve domain parameters over the prime finite field F_p can be described by one sextuple: *T=(p,a,b,G,n,h)*

'*p*': the prime number that defines the field and at the same time decides the curve form;

'a' and *'b'*: define the curve $y^2 \bmod p = x^3 + ax + b \bmod p$ and is chosen depending on the security requirement;

G: the generator point, $G = (x_G, y_G)$, one element in $E(F_p)$, which has the largest order *n*;

n: the order of *G,* large prime; Order of a point *G* on the curve can be defined as a value *n* such that

$nG = G+G+....upto\ n\ times = O$(Point at infinity).

'h': if $\#E(F_p)$ is the number of points on an elliptic curve then *'h'* is the cofactor where $h=\#E(F_p)/n$.

VIII. ELGAMAL ELLIPTIC CURVE CRYPTOSYSTEM

Generally we have two entities in the process of encryption and decryption. The one at the encryption side(Let us assume: Alice) and the other at the decryption side(Let us assume: Bob) Now Alice wants to send a message to Bob securely. The implementation of the ElGamal Elliptic Curve Cryptosystem can be divided into seven different phases which are explained below.

A. System Setup:

Setting up an elliptic curve cryptosystem requires both the entities to select and agree upon

a) the same set of ECC domain parameters, *i.e.* the underlying finite field F_p, the elliptic curve coefficients '*a*' and *'b'* for an appropriate elliptic curve, the generator point *G* in the curve, the order of *G i.e. n* and the cofactor $h=\#E(F_p)/n$.

b) the same set of algorithms for representing the elliptic curve points and implementing the elliptic curve operations in the chosen finite field.

B. Key Generation:

The next step is to generate the public and private key pair by both entities.

Algorithm for key generation:

a) Select a random number *k* from the interval*[1,n-1];*

b) Compute *Q=k*G;*

'k' is the private key and '*Q'* is the public key.

Suppose the private key and public key pair of Alice is (k_a,Q_a) and the same of Bob is (k_b,Q_b).

C. Public Key Validation

When receiving other's public key(say *Q*), the entity needs to take the following steps to validate the public key's legitimacy.

a) Check that $Q \neq O$, the point at infinity ;

b) Check that $(x_Q,y_Q) \in F_p$, where x_Q and y_Q denote the *x*-coordinate and *y*-coordinate of point *Q*.

c) Check that *Q* lies on the elliptic curve defined by *a* and *b*;

d) Check that $n*Q=O$, the point at infinity.
(as $n*Q=n*k*G=k*n*G=k*O=O$ where *G*'s order is *n*)

The public key validation without Step 3 is called the *partial* public-key validation. Without Step 4, the entity could be attacked. However, we can carefully select *h* to reduce the threat.

D. Message Encoding (Representation of Message into Elliptic Curve Point):

ECC Encryption and Decryption methods can only encrypt and decrypt a point on the curve not messages. Unfortunately, there is no known polynomial time algorithm 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 $E_p(a,b)$ 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. Let us suppose a text file has to be encrypted. 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(due to 8-bits in the ASCII character), 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 the easiest method for embedding a message but less efficient in terms of security.

The method proposed by Koblitz represents a message as a point or a set of points on an elliptic curve. We used this method for message encoding which is given bellow:

Suppose *E* is an elliptic curve given by $y^2 \bmod p = x^3 + ax + b \bmod p$ over a field F_p where *p* is a large prime. Using the following steps a message can be mapped to elliptic curve points.

a) Divide the message into blocks of fixed size. Express each block as number *x*. (For example, if the 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. This converts the message into a series of numbers between 0 and 35 where the block size is 1 character. We can also take the ASCII values of the characters.)

b) Compute $\alpha=x^3+ax+b \bmod p$.

c) Compute $\delta=\alpha^{(p-1)/2} \bmod p$.

d) If $\delta \neq 1$ set $x = x + 1$, go to step b).

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

e) Compute the square root β of $\alpha \bmod p$ (We used Las Vegas Algorithm);
f) Set $y=\beta$.
g) The point $P_m=(x, y)$ represents the particular block of the message.

E. Encryption

a) Alice gets Bob's public key (Q_b).
b) Alice checks the validity of Bob's public key using the method discussed above.
c) Then she selects a random number r where $r \in [1,n-1]$ compute $C_1 = r*G$ where G is the generator point.
d) Compute $C_2 = r*Q_b + P_m$, where P_m is the output elliptic curve point from message encoding algorithm given above.

$C=(C_1,C_2)$ is the ciphertext pair of points for the plaintext point P_m and is sent to Bob.

F. Decryption

a) Bob computes $M=k_b*C_1$ where k_b is his private key.
b) Computes $P_m = C_2 - M$, because $C_2 - M=C_2-k_b*C_1 = r*Q_b + P_m - k_b*r*G = r*k_b*G + P_m - k_b*r*G=P_m$.
c) The point P_m is the representation of the message block.

G. Message Decoding(Representation of Elliptic Curve Point into Message):

a) Consider each point $P_m= (x,y)$ and set m to be the greatest integer less than x. Then the point (x,y) decodes as the symbol m.
b) Convert m to message.

IX. ALGORITHMS USED FOR C IMPLEMENTATION

A. Algorithm 1- Binary Inversion Algorithm: This algorithm computes the inverse of a non-zero field element *'a'* belongs to $[1,p-1]$ using a variant of the Extended Euclidean Algorithm (EEA)[12][8]. The algorithm maintains the invariants $Aa + dp = u$ and $Ca + ep = v$ for some d and e which are not explicitly computed. The algorithm terminates when $u = 0$, in which case $v = 1$ and $Ca + ep = 1$, hence $C = a^{-1} \bmod p$.

Input: Prime p, a∈ [1,p - 1]
Output: $a^{-1} \bmod p$

```
1.  u←a; v←p; A←1; C←0;
2.  While u≠0 do:
3.      While u is even do:
4.          u←u/2;
5.          If A is even then:A←A/2;
6.          Else:A ←(A + p)/2; EndIf;
7.      EndWhile;
8.      While v is even do:
9.          v←v/2;
10.         If C is even then:C←C/2;
11.         Else:C←(C + p)/2;  EndIf;
12.     EndWhile;
13.     If u≥v then: u←u – v; A ←A - C;
14.     Else: v ←v – u; C←C – A;
15.     EndIf;
16. EndWhile;
17. Return(C);
```

B. Algorithm 2- Barrett Reduction: In modular arithmetic, Barrett reduction [15] is a reduction algorithm introduced in 1986 by P.D. Barrett.

Input: b the "base" of the integers used, p, $k = floor(\log_b p) + 1$, $x: 0 \le x < b^{2k}$, $u = floor((b^{2k})/p)$
Output: x mod p

```
1.  q1←floor(x / b^(k-1));
2.  q2← q1 * u;
3.  q3← floor(q2 / b^(k+1));
4.  r1←x mod b^(k+1);
5.  r2←(q3 * p) mod b^(k+1);
6.  r← r1 - r2;
7.  If(r < 0) then: r=r + b^(k+1); EndIf;
8.  While(r ≥ p) do: r=r – p;
9.  EndWhile;
10. Return (r);
```

The arithmetic in Barrett reduction can be reduced by choosing b to be a power of 2.

C. Point Multiplication Algorithm: A particularly useful signed digit representation is the non-adjacent form (NAF) which has the property that no two consecutive coefficients k_i are nonzero. Every positive integer k has a unique NAF, denoted as NAF(k). Moreover, NAF(k) has the fewest non-zero coefficients of any signed digit representation of k, and can be efficiently computed using Algorithm 3 given below .

D. Algorithm 3- Computing the NAF of a positive integer[12]:

Input: A positive integer k.
Output: NAF(k).

```
1.  i←0;
2.  While k ≥ 1 do:
3.      If k is odd then: k_i ←2 - (k mod 4); k←k – k_i;
4.      Else: k_i←0;EndIf;
5.      Return(k_i);
6.      k←k/2; i←i+1;
7.  EndWhile;
```

Binary NAF method uses NAF(k) instead of the binary representation of k. It is known that the length of NAF(k) is at most one longer than the binary representation of k. Also, the average density of non-zero coefficients among all NAFs of length l is approximately 1/3.

E. Algorithm 4: Binary NAF method for point multiplication

Input: NAF(k) $= \sum_{i=0}^{l-1} k_i 2^i$, $P \in E(F_p)$ *a point in the curve, l the string length of the NAF representation of k*
Output: kP

```
1.  Q←O, the point at infinity.
2.  For i from l-1 to 0 do:
3.    Q←2Q;
4.    If k_i=1 then:Q← Q + P;  EndIf;
5.    If k_i= -1 then:Q←Q – P;  EndIf;
6.    i←i-1;
7.  EndFor;
8.  Return(Q);
```

F. Algorithm 5: To calculate square root modulo of a number.

```
int Sqrt(int a, prime p)
1.  If (p mod 4=3) then: return (a^((p+1)/4) mod p);
2.  Else:randomly choose b ∈ [0,p-1];
3.  EndIf;
4.  i←(p - 1)/2; j←0;
5.  Repeat:
6.      i←i/ 2; j← j/2;
7.      if (a^i*b^j= -1 mod p) then: j←j + (p -1)/2;
8.      EndIf;
9.  Until:i=1 mod 2;
10. Return ((a^((i+1)/2) )*(b^j/2)mod p);
```

This algorithm is called Las Vegas Algorithm [8].

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

G. *Algorithm 6: Probabilistic Primality Test:* To test a number is prime or not .The test is called 'Probabilistic Primality Test'[8] .If the below algorithm is iterated n times, it will produce a false prime with probability not greater than $1/4^n$ .Therefore, n≥50 will give an acceptable probability of error.

1. *Set i←1 and n≥ 50;*
2. *Set w←the integer to be tested;*
3. *Find m such that w=1+2^a.m; where m is odd and a is the largest power of 2 dividing w-1;*
4. *Generate a random number b in the range 1<b<w;*
5. *Set j←0 and z←b^m mod w;*
6. *If j=0 and z=1 or if z=w-1 then: go to step 12; EndIf;*
7. *j←j+1;*
8. *If (j<a) then: set z←z^2 mod w and go to step 6;*
9. *Else: w is not prime; Stop; EndIf;*
10. *If i<n then: set i←i+1 and go to step 4;*
11. *Else: w is probably prime;*
12. *EndIf;*
13.

H. *Algorithm 7:* To perform modular addition (eccfadd)

Input: - k, a, p a prime number
Output: - (k+a) mod p

1. *k←mod(k,p);*
2. *a←mod(a,p);*
3. *addp←mod(k+a,p);*
4. *return(addp);*

I. *Algorithm 8*- To perform modular multiplication:

Input: - k1, a, p a prime number
*Output: - (k1*a) mod p*

1. *If(k1<0) then:k1←mod(k1,p);EndIf ;*
2. *naf(k1,u,&l);*
3. *ka=0;*
4. *For i←l-1to 0:*
5. *ka=eccfadd(ka,ka,p);*
6. *If(*(u+i)=1) then: ka←eccfadd(ka,a,p);*
7. *EndIf;*
8. *If(*(u+i)=-1) then:*
9. *nega←mod(-a,p);*
10. *ka←eccfadd(ka,nega,p);*
11. *EndIf;*
12. *i←i-1;*
13. *EndFor;*
14. *Return(ka);*

X. ECC Domain Parameters used in the Implementation

Domain parameters generated by us and used by us are given below:-

The domain parameters in the table below are taken from Wan Khudri and Sutanto's paper[4].

Table1: Domain parameters taken from [4]

Parameters	Values
p	3946183951
a	537680305
b	1059676324
x_G	1152222263
y_G	3133703258
n	3946206427
h	1.1

And the domain parameters in the table below are generated by us.

Table 2: Domain parameters generated

Parameters	Values
p	9463
a	1027
b	6584
x_G	4878
y_G	4444
n	9549
h	0.107

p=Prime field for the curve $y^2 mod\ p=x^3+ax+b\ mod\ p$.
a,b=curve coefficients.
$G(x_G,y_G)$=Generator point.
n=order of G.
h=cofactor of the curve.
The curve generated by us:-

$y^2 mod\ 9463=x^3+1027*x+6584\ mod\ 9463$

XI. Output

The sample plaintext file, cipher text files and decrypt file are given below:-

Plaintext.dat: `hello world`

Cipher1.dat (for the parameters in Table 1.)

```
*******THECIPHERTEXT IS********
3087908772 1436578608 343894993 2175666516 953400124
3655678255 548165374 886689848 2971263888 19501384
2173799023 3059246152 1210705334 60351030 1821880351
2516468027
```

Cipher2.dat (for the parameters in Table 2.)

```
*******THECIPHERTEXT IS********
3488 3748 4735 4328 2730 585 2857 2960 6815 8641 2158 4474
6800 3243 174 7242
```

Decrypt.dat

```
*****THE DECYRPTED PLAINTEXT IS ****
hello world
```

XII. Conclusion and Future Work

We have described an implementation of ElGamal Elliptic Curve Cryptosystem in C language. Here we are performing encryption of the plaintext and the encrypted message(i.e. ciphertext) is sent to the intended receiver, so that the secrecy of the message is maintained. The receiver then performs decryption of the ciphertext to get the plaintext.

This proposed design approach can also be used in the hardware implementation of Elliptic Curve Cryptography by using the elliptic curve generated. Future work includes the VLSI implementation of ElGamal ECC Encryption Scheme and to use the Elliptic Curve Cryptography in image encryption.

References

[1] V. S. Miller, "Use of Elliptic Curves in Cryptography," Advances in Cryptology, Vol. 218, pp. 417-426 ,1985

[2] N. Koblitz, "Elliptic Curve Cryptosystems," Mathematics of Computation, Vol. 48, pp. 203-209, 1987

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," Communications of the ACM, 21(2):120-126, February 1978
[4] Wan Khudri and Sutanto, "Implementation of Elgamal ECC using MatLab," International Conference on Instrumentation Communication and Information Technology(ICCI) 2005 Proc, August 3rd -5th ,2005, Bandung,Indonesia
[5] Sarwono Sutikno, Andy Surya, Ronny Effendi, "Implementation of ElGamal Elliptic Curves Cryptosystems," IEEE,1998.
[6] "Standards for Efficient Cryptography, SEC 1: Elliptic Curve Cryptography," Certicom Research, Sept 20,2000 Version1.0
[7] "Standards for Efficient Cryptography, SEC 2: Recommended Elliptic Curve Domain Parameters," September 20,2000.Version 1.0
[8] Menezes, J., Van Oorschot, P. C., and Vanstone, S. A., "Handbook of Applied Cryptology" CRC Press, LLC 1997.
[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] Oswald Elisabeth, "Introduction to Elliptic Curve Cryptography," Institute for Applied Information Processing and Communication, A-8010 Inffeldgasse16a , Graz, Austria, July 29, 2005.
[11] Maley O' Amiee, "Elliptic Curves and Elliptic Curve Cryptography," Undergraduate Colloquium Series
[12] Brown M, Hankerson D, Lopez J, and Menezes A, "Software Implementation of Elliptic Curves over Prime Fields".
[13] Marsaglia George, "Random Number Generators," Journal of Modern Applied Statistical Methods,May,2003,Vol 2,No 1,2 -13.
[14] Barrett, P., "Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor," Proceedings CRYPTO'86, pp. 311-323.
[15] M. Johnson, B. Phung, T. Shackelford, S. Rueangvivatanakij, "Modular Reduction of Large Integers Using Classical, Barrett, Montgomery Algorithms".
[16] 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.

ISBN No.978-1-4799-3834-6/14/$31.00©2014 IEEE