Elliptic Curve Cryptography

An Implementation Guide

Anoop MS
anoopms@tataelxsi.co.in

Abstract: The paper gives an introduction to elliptic curve cryptography (ECC) and how it is used in the implementation of digital signature (ECDSA) and key agreement (ECDH) Algorithms. The paper discusses the implementation of ECC on two finite fields, prime field and binary field. It also gives an overview of ECC implementation on different coordinate systems called the projective coordinate systems. The paper also discusses the basics of prime and binary field arithmetic.

1. Introduction

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.

The mathematical operations of ECC is defined over the elliptic curve $\mathbf{y^2 = x^3 + ax + b}$, where $\mathbf{4a^3 + 27b^2 \neq 0}$. Each value of the 'a' and 'b' gives a different elliptic curve. All points (x, y) which satisfies the above equation plus a point at infinity lies on the elliptic curve. The public key is a point in the curve and the private key is a random number. The public key is obtained by multiplying the private key with the generator point G in the curve. The generator point G, the curve parameters 'a' and 'b', together with few more constants constitutes the domain parameter of ECC. The EC domain parameters are explained in section 9.

One main advantage of ECC is its small key size. A 160-bit key in ECC is considered to be as secured as 1024-bit key in RSA.

2. 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. k is the discrete logarithm of Q to the base P.

Hence the main operation involved in ECC is point multiplication. i.e. multiplication of a scalar k with any point P on the curve to obtain another point Q on the curve.

3. Point multiplication

In point multiplication a point P on the elliptic curve is multiplied with a scalar k using elliptic curve equation to obtain another point Q on the same elliptic curve.

i.e. kP=Q
Point multiplication is achieved by two basic elliptic curve operations

- Point addition, adding two points J and K to obtain another point L i.e., L = J + K.
- Point doubling, adding a point J to itself to obtain another point L i.e. L = 2J.

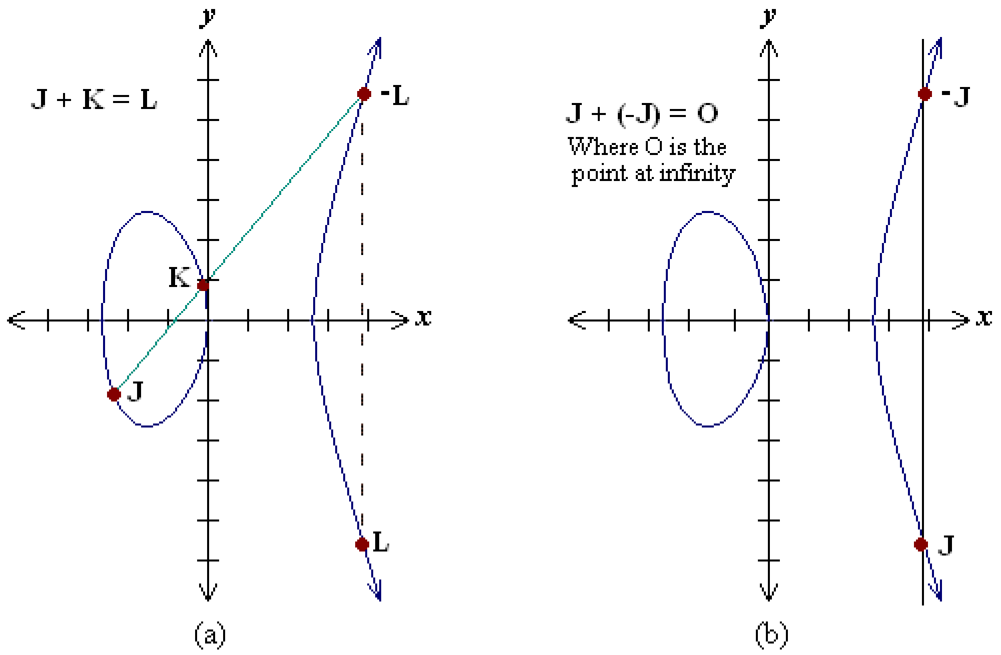
Point addition and doubling are explained in sections 4 and 5 respectively

Here is a simple example of point multiplication.
Let P be a point on an elliptic curve. Let k be a scalar that is multiplied with the point P to obtain another point Q on the curve. i.e. to find Q = kP.
If k = 23 then kP = 23.P = 2(2(2(2P) + P) + P) + P.
Thus point multiplication uses point addition and point doubling repeatedly to find the result. The above method is called 'double and add' method for point multiplication. There are other efficient methods for point multiplication such as NAF (Non – Adjacent Form) and wNAF (windowed NAF) method for point multiplication [1].

4. Point addition

Point addition is the addition of two points J and K on an elliptic curve to obtain another point L on the same elliptic curve.

4.1. Geometrical explanation



Consider two points J and K on an elliptic curve as shown in figure (a). If K ≠ -J then a line drawn through the points J and K will intersect the elliptic curve at exactly one more point –L. The reflection of the point –L with respect to x-axis gives the point L, which is the result of addition of points J and K.
Thus on an elliptic curve L = J + K.
If K = -J the line through this point intersect at a point at infinity O. Hence J + (-J) = O. This is shown in figure (b). O is the additive identity of the elliptic curve group.
A negative of a point is the reflection of that point with respect to x-axis.

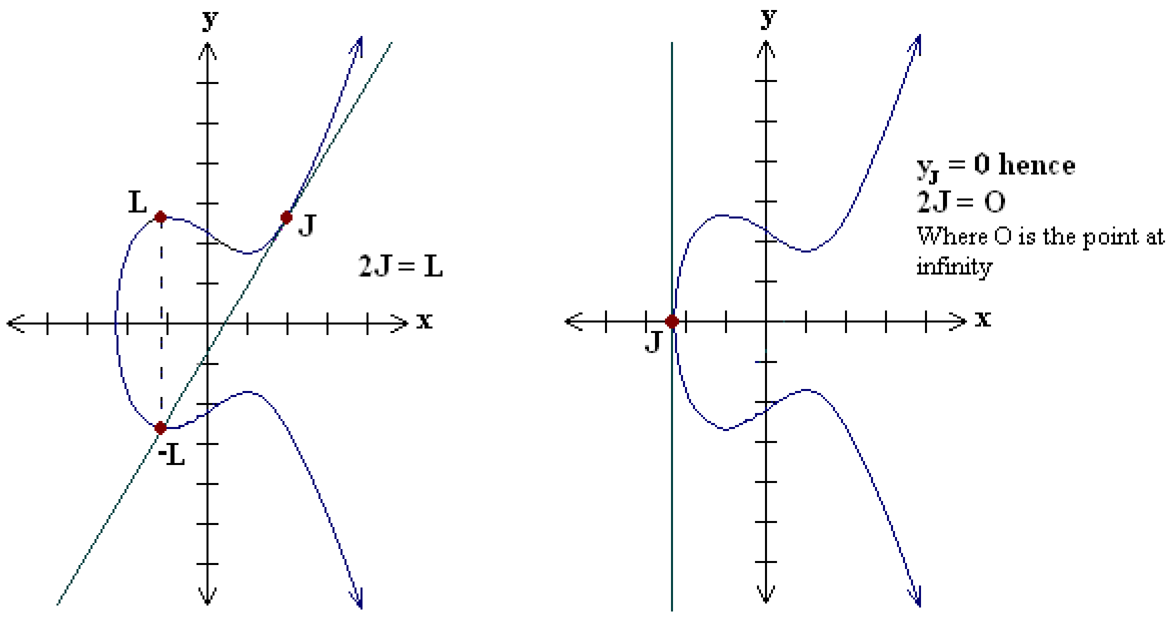
4.2. Analytical explanation

Consider two distinct points J and K such that J = (x_J, y_J) and K = (x_K, y_K)
Let L = J + K where L = (x_L, y_L), then
$x_L = s^2 - x_J - x_K$
$y_L = -y_J + s (x_J - x_L)$
$s = (y_J - y_K)/(x_J - x_K)$, s is the slope of the line through J and K.
If K = -J i.e. K = $(x_J, -y_J)$ then J + K = O. where O is the point at infinity.
If K = J then J + K = 2J then point doubling equations are used.
Also J + K = K + J

5. Point doubling

Point doubling is the addition of a point J on the elliptic curve to itself to obtain another point L on the same elliptic curve.

5.1. Geometrical explanation



(a) (b)

To double a point J to get L, i.e. to find L = 2J, consider a point J on an elliptic curve as shown in figure (a). If y coordinate of the point J is not zero then the tangent line at J will intersect the elliptic curve at exactly one more point –L. The reflection of the point –L with respect to x-axis gives the point L, which is the result of doubling the point J.
Thus L = 2J.
If y coordinate of the point J is zero then the tangent at this point intersects at a point at infinity O. Hence 2J = O when $y_J = 0$. This is shown in figure (b).

5.2. Analytical explanation

Consider a point J such that J = (x_J, y_J), where $y_J \neq 0$
Let L = 2J where L = (x_L, y_L), Then
$x_L = s^2 - 2x_J$
$y_L = -y_J + s(x_J - x_L)$

$s = (3x_J^2 + a) / (2y_J)$, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve
If $y_J = 0$ then $2J = O$, where O is the point at infinity.

6. Finite Fields

The elliptic curve operations defined above are on real numbers. Operations over the real numbers are slow and inaccurate due to round-off error. Cryptographic operations need to be faster and accurate. 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. Section 7 and 8 explains the EC operations on finite fields. The operations in these sections are defined on affine coordinate system. 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).

7. EC 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 **$4a^3 + 27b^2 \bmod p \neq 0$**. Here the elements of the finite field are integers between 0 and p – 1. All the operations such as addition, substation, division, multiplication involves integers between 0 and p – 1. This is modular arithmetic and is defined in session 10.1. 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 specifies curves with p ranging between 112-521 bits [4].
The graph for this elliptic curve equation is not a smooth curve. Hence the geometrical explanation of point addition and doubling as in real numbers will not work here. However, the algebraic rules for point addition and point doubling can be adapted for elliptic curves over $F_{p.}$

7.1. Point Addition

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$
Let L = J + K where $L = (x_L, y_L)$, then
$x_L = s^2 - x_J – x_K \bmod p$
$y_L = -y_J + s (x_J – x_L) \bmod p$
$s = (y_J – y_K)/(x_J – x_K) \bmod p$, s is the slope of the line through J and K.
If K = -J i.e. $K = (x_J, -y_J \bmod p)$ then J + K = O. where O is the point at infinity.
If K = J then J + K = 2J then point doubling equations are used.
Also J + K = K + J

7.2. Point Subtraction

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$
Then J - K = J + (-K) where $-K = (x_k, -y_k \bmod p)$
Point subtraction is used in certain implementation of point multiplication such as NAF [1].

7.3. Point Doubling

Consider a point J such that $J = (x_J, y_J)$, where $y_J \neq 0$
Let L = 2J where $L = (x_L, y_L)$, Then
$x_{L\,=}\ s^2 – 2x_J \bmod p$
$y_{L\,=}\ -y_J + s(x_J - x_L) \bmod p$

$s = (3x_J^2 + a) / (2y_J) \bmod p$, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve
If $y_J = 0$ then $2J = O$, where O is the point at infinity.

8. EC on Binary field F_2^m

The equation of the elliptic curve on a binary field F_2^m is $\mathbf{y^2 + xy = x^3 + ax^2 + b}$**,** where $\mathbf{b \neq 0}$. Here the elements of the finite field are integers of length at most m bits. These numbers can be considered as a binary polynomial of degree m – 1. In binary polynomial the coefficients can only be 0 or 1. All the operation such as addition, substation, division, multiplication involves polynomials of degree m – 1 or lesser. The polynomial arithmetic is defined in session 10.2. The m is chosen such that there is finitely large number of points on the elliptic curve to make the cryptosystem secure. SEC specifies curves with m ranging between 113-571 bits [4].
The graph for this equation is not a smooth curve. Hence the geometrical explanation of point addition and doubling as in real numbers will not work here. However, the algebraic rules for point addition and point doubling can be adapted for elliptic curves over F_2^m.

8.1. Point Addition

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$
Let $L = J + K$ where $L = (x_L, y_L)$, then
$x_L = s^2 + s + x_J + x_K + a$
$y_L = s (x_J + x_L) + x_L + y_J$
$s = (y_J + y_K)/(x_J + x_K)$, s is the slope of the line through J and K.
If $K = -J$ i.e. $K = (x_J, x_J + y_J)$ then $J + K = O$. where O is the point at infinity.
If $K = J$ then $J + K = 2J$ then point doubling equations are used.
Also $J + K = K + J$

8.2. Point Subtraction

Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$
Then $J - K = J + (-K)$ where $-K = (x_k, x_k + y_k)$
Point subtraction is used in certain implementation of point multiplication such as NAF [1].

8.3. Point Doubling

Consider a point J such that $J = (x_J, y_J)$, where $x_J \neq 0$
Let $L = 2J$ where $L = (x_L, y_L)$, Then
$x_L = s^2 + s + a$
$y_L = x_J^2 + (s + 1)*x_L$
$s = x_J + y_J/ x_J$, s is the tangent at point J and a is one of the parameters chosen with the elliptic curve
If $x_J = 0$ then $2J = O$, where O is the point at infinity.

9. Elliptic Curve Domain parameters

Apart from the curve parameters a and b, there are other parameters that must be agreed by both parties involved in secured and trusted communication using ECC. These are domain parameters. The domain parameters for prime fields and binary fields are described below. The generation of domain parameters is out of scope of this paper. There are several standard domain parameters defined by SEC [4].
Generally the protocols implementing the ECC specify the domain parameters to be used.

9.1. Domain parameters for EC over field F_p

The domain parameters for Elliptic curve over F_p are **p**, **a**, **b**, **G**, **n** and **h**.
p is the prime number defined for finite field F_p . a and b are the parameters defining the curve $y^2 \bmod p= x^3 + ax + b \bmod p$. G is the generator point (x_G, y_G), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where h = $\#E(F_p)/n$. $\#E(F_p)$ is the number of points on an elliptic curve.

9.2. Domain parameters for EC over field $F_2{}^m$

The domain parameters for elliptic curve over $F_2{}^m$ are **m**, **f(x)**, **a**, **b**, **G**, **n** and **h**.
m is an integer defined for finite field $F_2{}^m$. The elements of the finite field $F_2{}^m$ are integers of length at most m bits. f(x) is the irreducible polynomial of degree m used for elliptic curve operations which is discussed in section 10.2. a and b are the parameters defining the curve $y^2 + xy = x^3 + ax^2 + b$. G is the generator point (x_G, y_G), a point on the elliptic curve chosen for cryptographic operations. n is the order of the elliptic curve. The scalar for point multiplication is chosen as a number between 0 and n – 1. h is the cofactor where h = $\#E(F_2{}^m)/n$. $\#E(F_2{}^m)$ is the number of points on an elliptic curve.

10. Field Arithmetic

ECC uses modular arithmetic or polynomial arithmetic for its operations depending on the field chosen. The arithmetic involves big numbers in the range of 100s of bits. This section gives a brief overview for these two finite field operations.

10.1. 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 in to the range 0 and p – 1.

Addition
Let p = 23, a = 15, b = 20
a + b (mod p) = 15 + 20 (mod 23) = 35 mod 23 = 12
Since the result of a + b = 35 which is out of the range [0 22], The result is wrapped around in to the range [0 22] by subtracting 35 with 23 till the result is in range [0 22].
a mod b is thus explained as remainder of division a/b.

Subtraction
Let p = 23, a = 15, b = 20
a - b (mod p) = 15 - 20 (mod 23) = -5 mod 23 = 18
Since the result of a - b = -5 which is negative and out of the range [0 22], The result is wrapped around in to the range [0 22] by adding -5 with 23 till the result is in range [0 22].

Multiplication
Let p = 23, a = 15, b = 20
a * b (mod p) = 15 * 20 (mod 23) = 300 mod 23 = 1
Since the result of a * b = 300 which is out of the range [0 22], The result is wrapped around in to the range [0 22] by subtracting 300 with 23 till the result is in range [0 22].

Division
The division a/b (mod p) is defined as a * b^{-1} (mod p). b^{-1} is the multiplicative inverse of b over p.

Multiplicative Inverse
Multiplicative inverse of number b with respect to mod p is defined as a number b^{-1} such that $b*b^{-1}$ (mod p) = 1. Multiplicative inverse exists only if b and n are relatively prime. The algorithm such as extended Euclidean algorithm [7] can be used to find the multiplicative inverse of a number efficiently. Finding multiplicative inverse is a costly operation.

Finding x mod y
x mod y is the remainder of the division x/y. Finding x mod 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 [7] can be used to find modulus of a number in efficient manner.

10.2. Polynomial Arithmetic

Elliptic curve over field F_2^m involves arithmetic of integer of length m bits. These numbers can be considered as binary polynomial of degree m – 1. The binary string (a_{m-1} ... a_1 a_0) can be expressed as polynomial $a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + ... + a_2x^2 + a_1x + a_0$ where a_i = 0 or 1. For e.g., a 4 bit number 1101_2 can be represented by polynomial as $x^3 + x^2 + 1$. Similar to the modulus p on modular arithmetic, there is an irreducible polynomial of degree m in polynomial arithmetic. If in any operation the degree of polynomial is greater than or equal to m, the result is reduced to a degree less than m using irreducible polynomial also called as reduction polynomial.
In binary polynomial the coefficients of the polynomial can be either 0 or 1. If in any operation the coefficient becomes greater than 1, it can be reduced to 0 or 1 by modulo 2 operation on the coefficient.
All the operations below are defined in field F_2^4 are on irreducible polynomial $f(x) = x^4 + x + 1$. Since m = 4 the operation involves polynomial of degree 3 or lesser.

Addition
Consider two polynomial $A = x^3 + x^2 + 1$ and $B = x^2 + x$. On polynomial addition A + B gives $x^3 + 2x^2 + x + 1$. Taking mod 2 over coefficients, $A + B = x^3 + x + 1$.
On binary representation
$A = 1101_2$
$B = 0110_2$
$A + B = 1011_2$ which is an XOR operation between A and B. This is true in all cases. Hence polynomial addition can be achieved by simple XOR of two numbers.
i.e. A + B = A XOR B

Subtraction
Addition and subtraction are same operation in F_2^m.
Consider two polynomial $A = x^3 + x^2 + 1$ and $B = x^2 + x$. On polynomial subtraction A - B gives $x^3 – x + 1$. Taking mod 2 over coefficients $A - B = x^3 + x + 1$
On binary representation
$A = 1101_2$
$B = 0110_2$
$A - B = 1011_2$ which is an XOR operation between A and B. This is true in all cases. Hence polynomial subtraction can be achieved by simple XOR of two numbers.
i.e. A - B = A XOR B

Multiplication
Consider two polynomial $A = x^3 + x^2 + 1$ and $B = x^2 + x$. On polynomial multiplication A * B gives $x^5 + x^3 + x^2 + x$. Coefficient are reduced to mod 2. Since m = 4 the results are to be reduces to a degree less than 4 by irreducible polynomial $x^4 + x + 1$.
i.e. $x^5 + x^3 + x^2 + x$ (mod f(x))
$= (x^4 + x + 1)x + x^5 + x^3 + x^2 + x$
$= 2x^5 + x^3 + 2x^2 + 2x$
$= x^3$, on reducing the coefficient on mod 2

On binary representation
$A = 1101_2$
$B = 0110_2$
$A * B = 1000_2$

Division

The division a/b(mod f(x)) is defined as a * b^{-1} (mod f(x)). b^{-1} is the multiplicative inverse of b over f(x).

Multiplicative Inverse

Multiplicative inverse of number b with respect to irreducible polynomial f(x) is defined as a number b^{-1} such that $b*b^{-1}$ (mod f(x)) = 1. The algorithm such as extended Euclidean algorithm can be used to find the multiplicative inverse of a polynomial efficiently. Finding multiplicative inverse is a costly operation.

Irreducible polynomial

Irreducible polynomial is an analogue to modulus p in modular arithmetic. Irreducible polynomial is a polynomial of degree m that cannot be expressed as the product of two polynomials of lesser degree.
If in any polynomial arithmetic operation the resultant polynomial is having degree greater than or equal to m, it is reduced to a polynomial of degree less than m by the irreducible polynomial. An example is shown in multiplication section above.
In many standard implementation of elliptic curve operation, for making polynomial reduction more efficient the irreducible polynomial is chosen to be trinomial (polynomial containing 3 terms) or pentanomial (polynomial containing 5 terms) [1].

11. EC Cryptography

The EC algorithms are specified in *SEC 1: Elliptic Curve Cryptography* [3]. An overview of EC cryptographic algorithms for key agreement and digital signature are explained below.

11.1. ECDSA - Elliptic Curve Digital Signature Algorithm

Signature algorithm is used for authenticating a device or a message sent by the device. For example consider two devices A and B. To authenticate a message sent by A, the device A signs the message using its private key. The device A sends the message and the signature to the device B. This signature can be verified only by using the public key of device A. Since the device B knows A's public key, it can verify whether the message is indeed send by A or not.
ECDSA is a variant of the Digital Signature Algorithm (DSA) that operates on elliptic curve groups. For sending a signed message from A to B, both have to agree up on Elliptic Curve domain parameters. The domain parameters are defined in section 9. Sender 'A' have a key pair consisting of a private key d_A (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key $Q_A = d_A * G$ (G is the generator point, an elliptic curve domain parameter). An overview of ECDSA process is defined below.

Signature Generation

For signing a message m by sender A, using A's private key d_A

1. Calculate e = HASH (m), where HASH is a cryptographic hash function, such as SHA-1
2. Select a random integer k from [1,n − 1]
3. Calculate r = x1 (mod n), where (x1, y1) = k * G. If r = 0, go to step 2
4. Calculate $s = k^{-1}(e + d_A r)$(mod n). If s = 0, go to step 2
5. The signature is the pair (r, s)

Signature Verification

For B to authenticate A's signature, B must have A’s public key Q_A

1. Verify that r and s are integers in [1,n − 1]. If not, the signature is invalid
2. Calculate e = HASH (m), where HASH is the same function used in the signature generation
3. Calculate $w = s^{-1} \pmod n$
4. Calculate $u_1 = ew \pmod n$ and $u_2 = rw \pmod n$
5. Calculate $(x1, y1) = u_1G + u_2Q_A$
6. The signature is valid if $x1 = r(\mathrm{mod}\ n)$, invalid otherwise

11.2. ECDH – Elliptic Curve Diffie Hellman

ECDH is a key agreement protocol that allows two parties to establish a shared secret key that can be used for private key algorithms. Both parties exchange some public information to each other. Using this public data and their own private data these parties calculates the shared secret. Any third party, who doesn’t have access to the private details of each device, will not be able to calculate the shared secret from the available public information. An overview of ECDH process is defined below.

For generating a shared secret between A and B using ECDH, both have to agree up on Elliptic Curve domain parameters. The domain parameters are defined in section 9. Both end have a key pair consisting of a private key d (a randomly selected integer less than n, where n is the order of the curve, an elliptic curve domain parameter) and a public key Q = d * G (G is the generator point, an elliptic curve domain parameter). Let (d_A, Q_A) be the private key - public key pair of A and (d_B, Q_B) be the private key - public key pair of B.

1. The end A computes $K = (x_K, y_K) = d_A * Q_B$
2. The end B computes $L = (x_L, y_L) = d_B * Q_A$
3. Since $d_AQ_B = d_Ad_BG = d_Bd_AG = d_BQ_A$. Therefore K = L and hence $x_K = x_L$
4. Hence the shared secret is x_K

Since it is practically impossible to find the private key d_A or d_B from the public key K or L, its not possible to obtain the shared secret for a third party.

12. To make it more efficient

As we discussed earlier the point multiplication is the main operation in elliptic curve cryptography. Point multiplication involves plenty of point addition and point doubling. Each point addition and doubling involves a multiplicative inverse operation each as seen in sections 7 and 8. Finding multiplicative inverse is a costly operation in both finite fields, F_p and $F_2{}^m$.

Representing the points in projective coordinate systems can eliminate the need of multiplicative inverse operation in point addition and point doubling and there by increasing the efficiency of point multiplication operation.

For using the projective coordinate in elliptic curve one has to convert the given point in affine coordinate to projective coordinate before point multiplication then convert it back to affine coordinate after point multiplication. The entire process requires only one multiplicative inverse operation. The operation in projective coordinate involves more scalar multiplication than in affine coordinate. ECC on projective coordinate will be efficient only when the implementation of scalar multiplication is much faster than multiplicative inverse operation.

13. EC operations in Projective coordinate system

EC on various projective coordinates have been proposed out of which one each for binary field and prime field are explained below.

13.1. Projective coordinate in field F_{2^m} [1]

Here the point (X, Y, Z) in projective coordinate corresponds to the point $(X/Z, Y/Z^2)$ in affine coordinate. The equation for the elliptic curve is $Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4$. For point multiplication, convert the point (X, Y) in affine coordinate to (X, Y, 1) in projective coordinate. After multiplication the result (X, Y, Z) is converted back to the affine coordinate as $(X/Z, Y/Z^2)$ where $Z \neq 0$. If Z = 0, then the point is considered as the point at infinity.

Point addition

For adding two points in projective coordinate Let $(X_1, Y_1, Z_1) + (X_2, Y_2, 1) = (X_3, Y_3, Z_3)$ then

$A = Y_2. Z_1^2 + Y_1$
$B = X_2. Z_1 + X_1$
$C = Z_1. B$
$D = B^2.(C + a. Z_1^2)$
$Z_3 = C^2$
$E = A.C$
$X_3 = A^2 + D + E$
$F = X_3 + X_2. Z_3$
$G = X_3 + Y_2. Z_3$
$Y_3 = E. F + Z_3.G$

$Z_2 = 1$, since one operand in point addition will always be the input point in point multiplication operation, which is an affine coordinate point.

Point doubling

For doubling a point in projective coordinate Let $2(X_1, Y_1, Z_1) = (X_3, Y_3, Z_3)$ then
$Z_3 = X_1^2. Z_1^2$
$X_3 = X_1^4 + b. Z_1^4$
$Y_3 = b. Z_1^4. Z_3 + X_3.(a. Z_3 + Y_1^2 + b.Z_1^4)$

13.2. Jacobian Projective coordinate in field F_p [2]

Here the point (X, Y, Z) in Jacobian projective coordinate corresponds to the point $(X/Z^2, Y/Z^3)$ in affine coordinate. The equation for the elliptic curve is $Y^2 = X^3 - 3.XZ^4 + bZ^6$. For point multiplication, convert the point (X, Y) in affine coordinate to (X, Y, 1) in Jacobian projective coordinate. After multiplication the result (X, Y, Z) is converted back to the affine coordinate as $(X/Z^2, Y/Z^3)$ where $Z \neq 0$. If Z = 0, then the point is considered as the point in infinity.

Point addition

For adding two points in projective coordinate Let $(X_1, Y_1, Z_1) + (X_2, Y_2, 1) = (X_3, Y_3, Z_3)$ then
$A = X_2. Z_1^2$
$B = Y_2. Z_1^3$
$C = A - X_1$
$D = B - Y_1$
$X_3 = D^2 - (C^3 + 2X_1.C^2)$
$Y_3 = D.(X_1.C^2 - X_3) - Y_1.C^3$
$Z_3 = Z_1. C$

Z_2 = 1, since one operand in point addition will always be the input point in point multiplication operation, which is an affine coordinate point.

Point doubling

For doubling a point in Jacobian projective coordinate Let $2(X_1, Y_1, Z_1) = (X_3, Y_3, Z_3)$ then

$A = 4X_1 + Y_1^2$

$B = 8Y_1^4$

$C = 3(X_1 - Z_1^2).(X1 + Z_1^2)$

$D = -2A + C^2$

$X_3 = D$

$Y_3 = C. (A - D) - B$

$Z_3 = 2Y_1.Z_1$

14. Conclusion

For efficient implementation of ECC, it is important for the point multiplication algorithm and the underlying field arithmetic to be efficient. There are different methods for efficient implementation point multiplication [1] and field arithmetic [1][7] suited for different hardware configurations.

Implementation of ECC using projective coordinates has shown considerable improvement in efficiency compared to the affine coordinate implementation. This improvement in efficiency is due to the elimination of multiplicative inverse operation in point addition and doubling that would otherwise cost considerable processor cycles.

If the irreducible polynomial in binary field implementation is chosen to be trinomial or pentanomial the implementation of ECC on binary field can be made efficient than the prime field implementation. In SEC specified domain parameters [4], the irreducible polynomials are either trinomial or pentanomial. These chosen polynomials cause the polynomial reduction in binary field to run much faster than the modular reduction in prime field.

Reference

[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] M. Brown, D. Hankerson, J. Lopez, A. Menezes, *Software Implementation of the NIST Elliptic Curves Over Prime Fields*, 2001, Available at http://citeseer.ist.psu.edu/brown01software.html

[3] 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

[4] 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

[5] Openssl, http://www.openssl.org

[6] Certicom, http://www.certicom.com/index.php?action=ecc_tutorial,home

[7] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996