ELLIPTIC CURVE CRYPTOGRAPHY

Public Key Cryptography

- Components
  - Public Key,
  - Private Key
  - Set of Operators that work on these Keys
  - Predefined Constraints (required by some algorithms)

Elliptic Curve Cryptography

- Components

Private Key	Public Key	Set of Operations	Domain Parameters (Predefined constants)
A random number	Point on a curve = Private Key * G	These are defined over the curve **$y^2 = x^3 + ax + b$,** where **$4a^3 + 27b^2 \neq 0$**	G, a, b

Discrete Logarithm Problem (DLP)

- Let P and Q be two points on the elliptic curve
  - Such that Q = kP, where k is a scalar value
- DLP: Given P and Q, find k?
  - If k is very large, it becomes computationally infeasible
- The security of ECC depends on the difficulty of DLP
- Main operation in ECC is Point Multiplication

Point Multiplication

- Point Multiplication is achieved by two basic curve operations:

1. Point Addition, L = J + K
2. Point Doubling, L = 2J

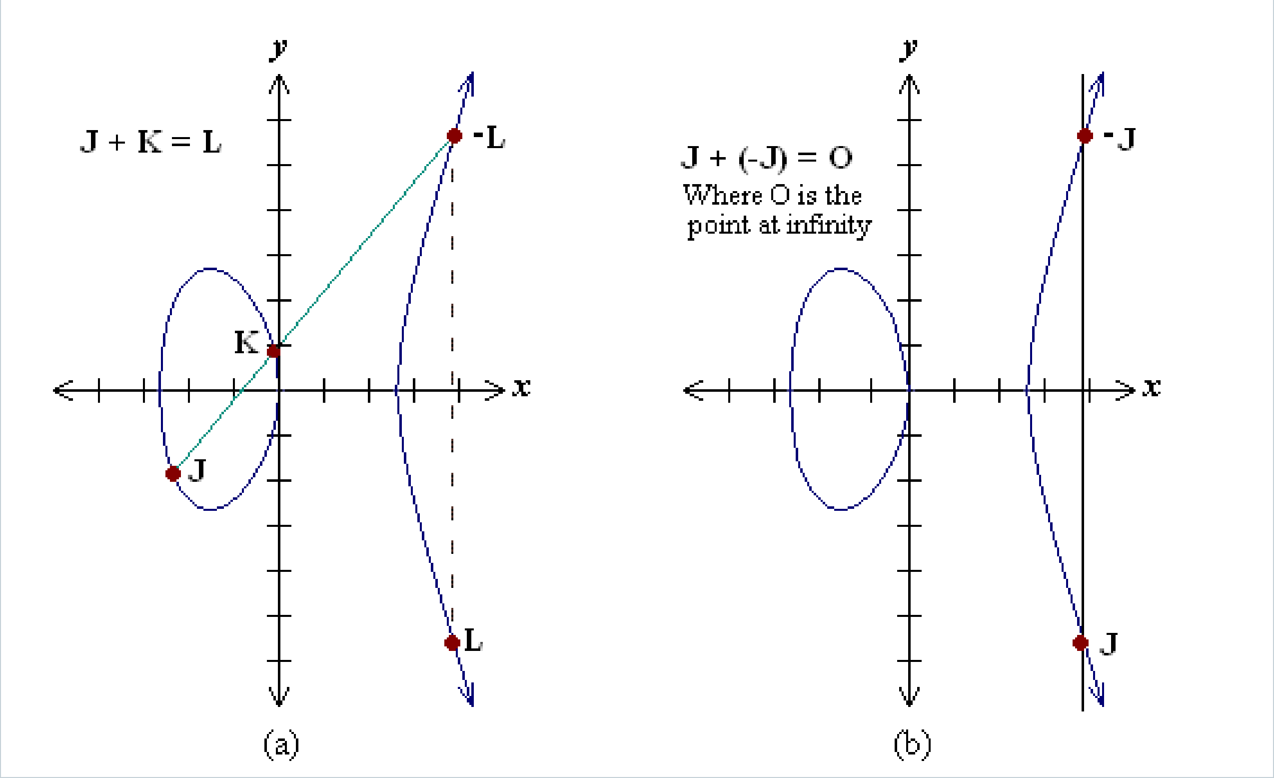
Example:

If k = 23;   then, kP = 23*P

$= 2(2(2(2P) + P) + P) + P$

Point Addition

Geometrical explanation:



Point Addition

- **Analytical explanation:**

- Consider two distinct points J and K such that $J = (x_J, y_J)$ and $K = (x_K, y_K)$

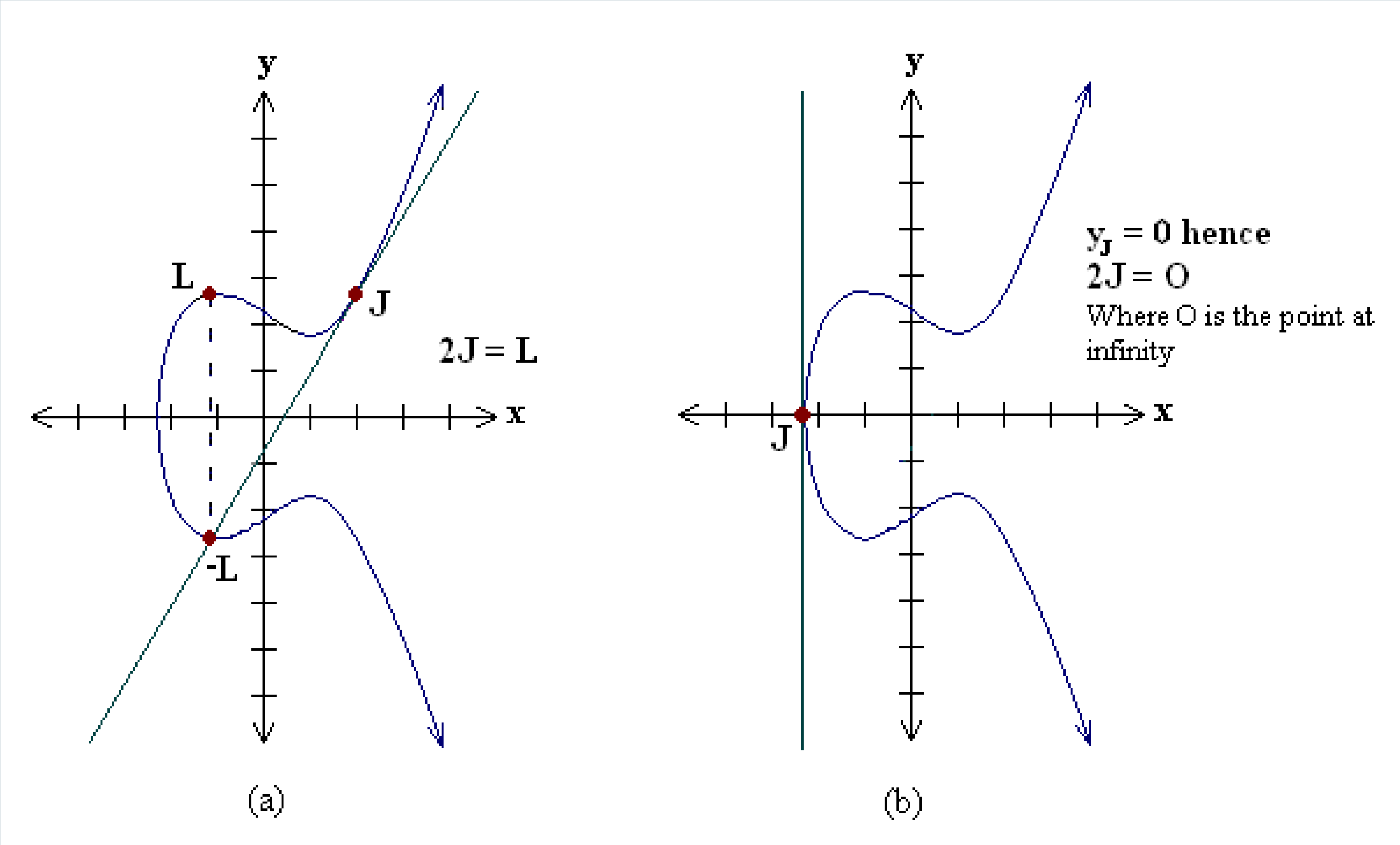
Let $L = J + K$ where $L = (x_L, y_L)$, then

- $x_L = s^2 - x_J - x_K$
- $y_L = -y_J + s\,(x_J - x_L)$

$s = (y_J - y_K)/(x_J - x_K)$, s is slope of the line through J and K

Point Doubling

Geometrical explanation:



Point Doubling

- **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

Finite Fields

- The Elliptic curve operations shown were on real numbers
  - Issue: operations are slow and inaccurate due to round-off errors

- To make operations more efficient and accurate, the curve is defined over two finite fields

1. Prime field F_p and
2. Binary field F_{2^m}

- The field is chosen with finitely large number of points suited for cryptographic operations

EC on Prime field F_p

- Elliptic Curve equation:

  $$y^2 \bmod p = x^3 + ax + b \bmod p$$

  where $4a^3 + 27b^2 \bmod p \neq 0$.
- Elements of finite fields are integers between 0 and p-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

EC on Binary field F_{2^m}

- Elliptic Curve equation:

  $$\mathbf{y^2 + xy = x^3 + ax^2 + b,}$$

  where $\mathbf{b \neq 0}$
- Here the elements of the finite field are integers of length at most **m** bits.
- In binary polynomial the coefficients can only be 0 or 1.
- 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

Elliptic Curve Domain parameters

Domain parameters for EC over field F_p

- Parameters:

  p, a, b, G, n and h.

Domain parameters for EC over field F_{2^m}

- Parameters:

  m, f(x), a, b, G, n and h.

Implementations

- **ECDSA - Elliptic Curve Digital Signature Algorithm**

Signature Generation:

For signing a message m by sender A, using A's private key d_A and public key $Q_A = d_A * G$

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 $r = x_1 \pmod n$, where $(x_1, y_1) = k * G$. If $r = 0$, go to step 2
4. Calculate $s = k^{-1}(e + d_A r) \pmod n$. If $s = 0$, go to step 2
5. The signature is the pair (r, s)

Implementations

- **ECDSA - Elliptic Curve Digital Signature Algorithm**

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 $(x_1, y_1) = u_1G + u_2Q_A$
6. The signature is valid if $x_1 = r \pmod n$, invalid otherwise

Implementations

- **ECDH – Elliptic Curve Diffie Hellman**

A $(Q_{A,}d_A)$ – Public, Private Key pair

B $(Q_{B,}d_B)$ – Public, Private Key pair

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