Blockchain

Nepal

Elliptic Curves Cryptography

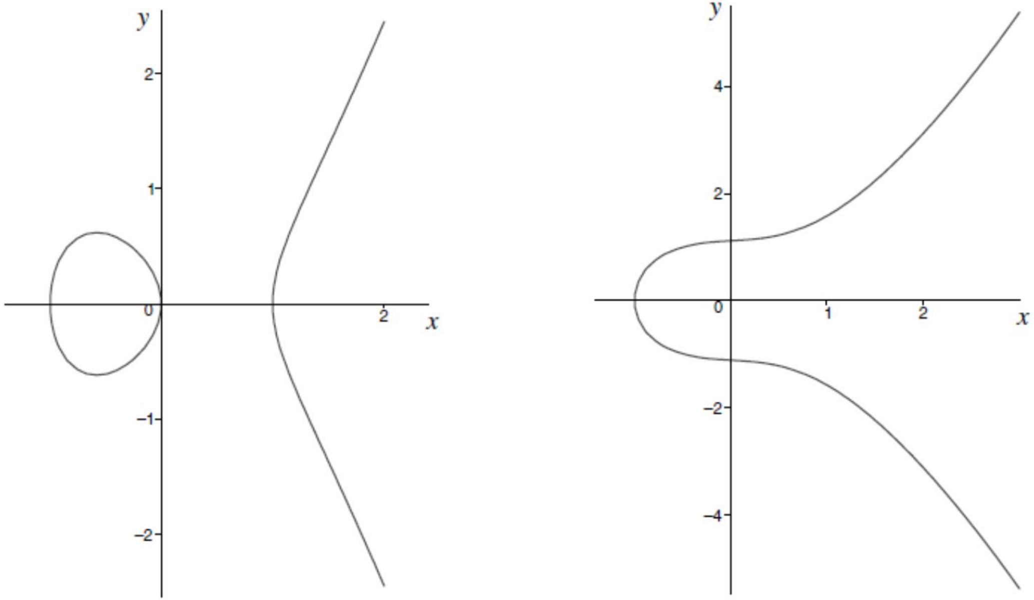
October 13, 2017

Elliptic Curves over real numbers

Plain curve with equation

$$y^2 = x^3 + a.x + b$$

(Weierstrass equation)



(a) $E_1 : y^2 = x^3 - x$ (b) $E_2 : y^2 = x^3 + \frac{1}{4}x + \frac{5}{4}$

Elliptic Curves over Finite Field

Elliptic Curve E: $y^2 = x^3 + a.x + b$ over finite field F_P where p is prime

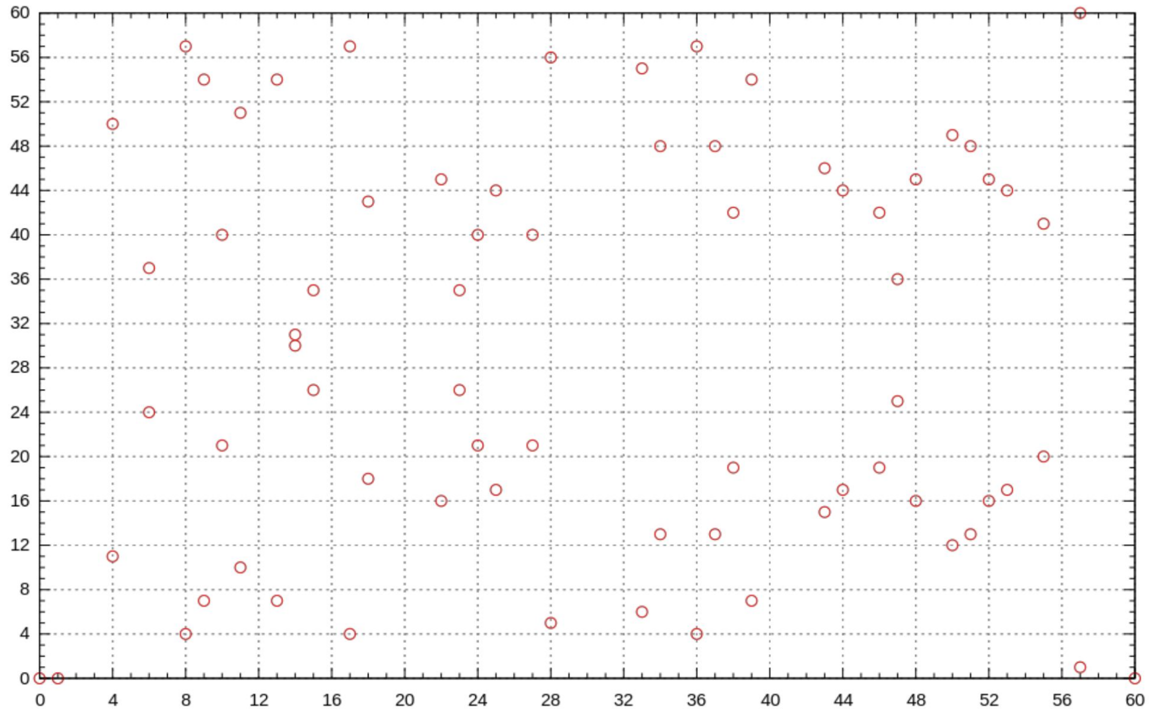
- $a,b \in F_P$ and $(x,y) \in F_P$ represents points on the curve
- There is a distinguished point called infinity ∞
- Set of all points on curve E is denoted by $E(F_P)$
- Example :
    - Let p = 7 and $y^2=x^3+2x+4$
    - $E(F_7) = \{ \infty,$ (0,2), (0,5), (1,0), (2,3), (2,4), (3,3),(3,4),(6,1),(6,6)$\}$

Example

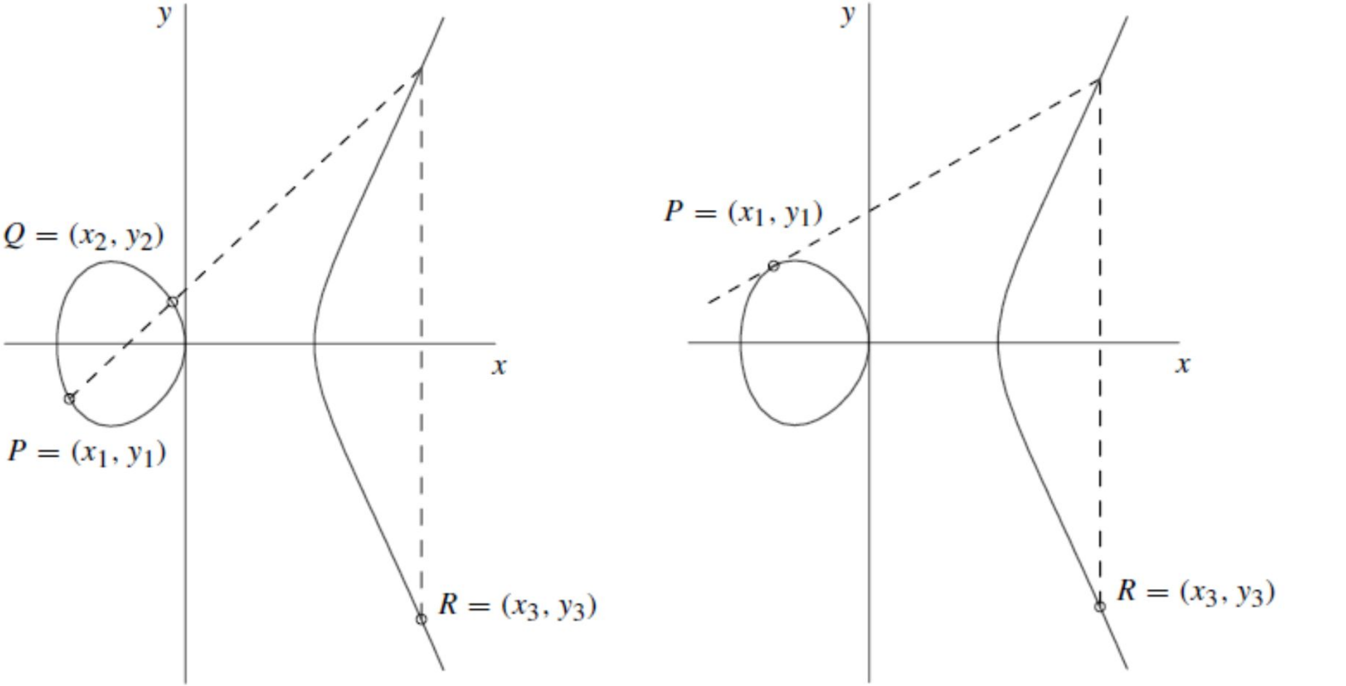
Elliptic Curve

$$\mathbf{y^2 = x^3 - x} \quad \text{over } F_{61}$$

Here, a = -1, b = 0



Elliptic Curve Operation - Addition



(a) Addition: $P + Q = R$.

(b) Doubling: $P + P = R$.

Elliptic Curve groups

Addition rules

Abelian Group

- Identity: P + inf. = inf + P = P
- Inverse: if P = (x,y), then -P = (x, -y), and P + (-P) = inf.
- Addition: P = (x_1, y_1) and Q = (x_2, y_2) and P != +/- Q, then P+Q = (x_3, y_3)

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2 \quad \text{and} \quad y_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)(x_1 - x_3) - y_1$$

- Doubling: P = (x_1, y_1) and P != -P, then 2P = (x_3, y_3)

$$x_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)^2 - 2x_1 \quad \text{and} \quad y_3 = \left(\frac{3x_1^2 + a}{2y_1}\right)(x_1 - x_3) - y_1$$

Cyclic subgroup of elliptic curve groups

Cyclic subgroup of $E(F_P)$ generated by a point P is

$$\{ \text{inf}, P, 2P, 3P, \ldots, (n-1)P\}$$

Such cyclic subgroups can be used to implement discrete logarithm systems

Elliptic Curve Discrete Log Problem (ECDLP)

Given an elliptic curve group $E(F_P)$, a generator P of cyclic subgroup of prime order n of $E(F_P)$, and a point Q in that subgroup, find the integer d, $1 <= d <= n-1$, such that $dP = Q$

ElGamal over elliptic curves

Key Generation

- Domain parameters
  - Prime p
  - Elliptic curve: E (eg. $y^2=x^3-x$)
  - Generator point P of cyclic subgroup of $E(F_P)$
  - Prime order n of the subgroup
- Private Key: random $d \in [1, n-1]$
- Public Key: $Q = dP$

Encryption

-- Input: domain params (p, E, P, n)

-- Public key Q; message m

-- Represent m as a point M in $E(F_P)$

-- Choose random $k \in [1, n-1]$

-- Compute $C_1=kP$ and $C_2=M+kQ$

-- Output: (C_1, C_2)

Decryption

-- Input: domain params (p, E, P, n); Private key d; Ciphertext (C_1, C_2)

-- Compute $C_2 - dC_1 = M + kQ - dkP = M + kdP - dkP = M$

-- Output : extract m from M

Why to choose elliptic curve crypto?

- Same level of security for smaller parameters in ECC

	Security level (bits)				
	80 (SKIPJACK)	112 (Triple-DES)	128 (AES-Small)	192 (AES-Medium)	256 (AES-Large)
DL parameter q EC parameter n	160	224	256	384	512
RSA modulus n DL modulus p	1024	2048	3072	8192	15360

- Faster operations

➡ private key operations for ECC many times efficient than RSA & DL private key operations

➡ Public key operations for ECC many times more efficient than those for DL systems

Standardized curves

- NIST (National Institute of Standards and Technology) curves
- SECG (Standards for Efficient Cryptography Group) curves
- ECC Brainpool curves

Popular curves:

➜ Secp256k1 (used in Bitcoin & other cryptocurrencies)

➜ Curve25519

ECC Example - Java

```
KeyPairGenerator generator = KeyPairGenerator.getInstance(" EC","BC");

generator.initialize(new ECGenParameterSpec(" secp256r1"));

KeyPair keypair = generator.genKeyPair();


// Public key
PublicKey pubKey = keyPair.getPublic();

// Private key
PrivateKey privateKey = keyPair.getPrivate();
```

ECC Example - Java

```java
// Encryption
String ALGORITHM = "ECIES";
Cipher cipher = Cipher.getInstance(ALGORITHM, "BC");
cipher.init(Cipher.ENCRYPT_MODE, publicKey);
byte[] cipherText = cipher.doFinal(message.getBytes());
```

```java
// Decryption
String ALGORITHM = "ECIES";
Cipher cipher = Cipher.getInstance(ALGORITHM, "BC");
cipher.init(Cipher.DECRYPT_MODE, privateKey);
byte[] plainText = cipher.doFinal(cipherText);
```

https://gist.github.com/anonymous/1c3eedb88b4294e16b451bc53d79f096

Exercise

1. Encrypt the following text with Curve25519 and decrypt the ciphertext to verify the encryption is correct.
   `“A quick brown fox jumps over the lazy dog”`
2. Serialize and store the public & private keys in file.