

# ECC (Elliptic Curve Cryptography)

ECC (Elliptic Curve Cryptography) is a public-key encryption algorithm used to secure data transmission over networks.

ECC is based on the mathematical concept of elliptic curves and it offers the same level of security as RSA and other public-key encryption algorithms, but with much shorter key lengths.

The ECC algorithm involves the following steps:

## 1. Key Generation:

- Choose a random elliptic curve over a finite field of prime order
- Select a point  $P$  on the elliptic curve as the base point
- Choose a private key (a random integer)  $d$
- Calculate the public key (a point on the elliptic curve)  $Q = dP$

Public key: The point  $Q$

Private key: The integer  $d$

## 2. Encryption:

- Choose a random integer  $k$
- Calculate the elliptic curve point  $C1 = kP$
- Calculate the shared secret  $s = kQ$
- Select a symmetric encryption algorithm (for example, AES)
- Encrypt the plaintext with the shared secret using the symmetric encryption algorithm
- Attach  $C1$  and the encrypted ciphertext to the message

## 3. Decryption:

- Given  $C1$ , calculate  $s = dC1$
- Decrypt the ciphertext using the shared secret  $s$

The mathematical formulas used in ECC algorithm:

- Elliptic Curves: An elliptic curve is a curve that satisfies a specific mathematical equation. In ECC, the curve is defined over a finite field of prime order.
- Base Point ( $P$ ): A point on the elliptic curve that is used as the basis for the elliptic curve calculations.
- Point Addition: ECC uses a point addition operation for addition of two points on the curve.
- Scalar Multiplication: ECC uses scalar multiplication for generation of public key from private key.

Example of JavaScript code for ECC Encryption and Decryption:

```

const crypto = require('crypto');
const eccrypto = require('eccrypto');

// ECC Key Generation
async function eccKeyGen() {
  const { publicKey, privateKey } = await eccrypto.generateKeypair();
  return { publicKey, privateKey };
}

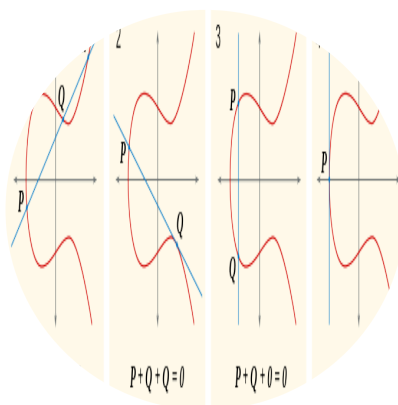
// ECC Encryption
async function eccEncrypt(publicKey, plaintextMsg) {
  const plaintextBuffer = Buffer.from(plaintextMsg, 'utf8');
  const k = crypto.randomBytes(32); // Generate random key
  const ciphertext = await eccrypto.encrypt(publicKey, plaintextBuffer, [{ k }]);
  return ciphertext;
}

// ECC Decryption
async function eccDecrypt(privateKey, ciphertextMsg) {
  const decrypted = await eccrypto.decrypt(privateKey, ciphertextMsg);
  return decrypted.toString('utf8');
}

```

## Note:

This implementation of ECC (Elliptic Curve Cryptography) formula.



Fun

$$y^2 = x^3 + 3x + b \pmod{p}$$

- $p$  = the hours on the clock (generate by  $\phi p$  &  $\phi q$ ) " $\phi = e = \text{euler}$ , read my article 04-phi-euler.md in same repo"
- $b$  = random big num

- $x, y$  = prime number ( $\phi p$  &  $\phi q$ )
- $\phi$ :  $p, q$  = have  $x$  &  $y$  axis on the curve (they are use in  $\phi$  for generating randoms)

this is the ecc revers engineering!

in fact you can create a 4096 encryption machine less then 100\$

**Conjectur:** elliptic curve is a 2d of inside the torus. explain that to your-self by imagin the shape in 3dimensions!

This conjecture is a *theorem* asap!

The way to create un-crackable cryptography...