

# Lesson 15: Public Key Cryptography & Digital Signatures

## Module 2: Blockchain Fundamentals

Digital Finance

# The Problem: Secure Communication Over Insecure Channels

## Challenge:

- How do two parties communicate securely without meeting?
- How do you verify someone's identity online?
- How do you prove authorship of a message?

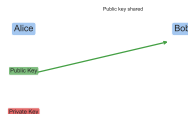
## Traditional Solution:

- Symmetric cryptography (shared secret)
- Problem: Key distribution
- Requires secure channel to share key

Symmetric Encryption



Asymmetric Encryption



# Symmetric vs Asymmetric Cryptography

[charts/lesson\\_15/crypto\\_comparison.pdf](#)

## Key Pair Structure:

- **Public Key:** Shared openly
- **Private Key:** Kept secret
- Mathematical relationship
- One-way function (easy to compute, hard to reverse)

## Properties:

- Encrypt with public → decrypt with private
- Sign with private → verify with public
- Cannot derive private from public

`charts/lesson_15/key_pair_generation.pdf`

## One-Way Function:

$$y = f(x) \quad (\text{easy})$$

$$x = f^{-1}(y) \quad (\text{hard})$$

## Examples:

- Factoring large primes (RSA)
- Discrete logarithm (Diffie-Hellman)
- Elliptic curve discrete log (ECDSA)

[charts/lesson\\_15/trapdoor\\_function.pdf](#)

**Trapdoor:** Secret information (private key) makes inverse easy

## Key Generation:

- 1 Choose two large primes:  $p, q$
- 2 Compute  $n = p \times q$  (modulus)
- 3 Compute  $\phi(n) = (p - 1)(q - 1)$
- 4 Choose public exponent  $e$  (commonly 65537)
- 5 Compute private exponent  $d \equiv e^{-1} \pmod{\phi(n)}$

## Encryption/Decryption:

Ciphertext:  $c = m^e \bmod n$  | Plaintext:  $m = c^d \bmod n$

**Example:**  $p = 61, q = 53, n = 3233, e = 17, d = 2753$

- Message  $m = 123$ :  $c = 123^{17} \bmod 3233 = 855$
- Decrypt:  $m = 855^{2753} \bmod 3233 = 123$

# Elliptic Curve Cryptography (ECC)

## Why ECC?

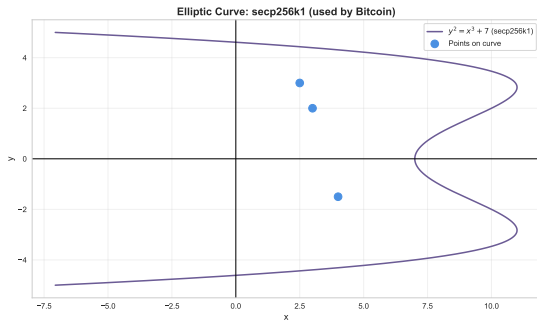
- Smaller key sizes (256-bit ECC  $\approx$  3072-bit RSA)
- Faster computations
- Lower bandwidth
- Standard in Bitcoin/Ethereum

## Curve Equation:

$$y^2 = x^3 + ax + b$$

Bitcoin uses: secp256k1

$$y^2 = x^3 + 7$$



## ECC Point Addition: The Core Operation

`charts/lesson_15/ecc_point_addition.pdf`



## Easy Problem:

- Given point  $G$  and scalar  $k$
- Compute  $P = k \cdot G$
- Fast using double-and-add

## Hard Problem (ECDLP):

- Given points  $G$  and  $P$
- Find scalar  $k$  such that  $P = k \cdot G$
- No efficient algorithm known
- This is the **private key**

[charts/lesson\\_15/ecdlp\\_visualization.pdf](#)

**Security:** Best attack takes  $\mathcal{O}(\sqrt{n})$  operations for  $n$ -bit key

## Purpose:

- Prove message was created by you
- Ensure message wasn't altered
- Non-repudiation (can't deny)

## Process:

- ① Hash the message
- ② Encrypt hash with private key
- ③ Attach signature to message
- ④ Verify: Decrypt with public key, compare hash

charts/lesson\_15/digital\_signature\_flow.pdf

## Signature Generation:

- 1 Message  $m$ , private key  $d$ , public key  $Q = d \cdot G$
- 2 Hash message:  $z = \text{hash}(m)$
- 3 Choose random  $k$ , compute  $R = k \cdot G = (x_R, y_R)$
- 4 Compute  $r = x_R \bmod n$
- 5 Compute  $s = k^{-1}(z + rd) \bmod n$
- 6 Signature:  $(r, s)$

## Signature Verification:

- 1 Compute  $w = s^{-1} \bmod n$
- 2 Compute  $u_1 = zw \bmod n$ ,  $u_2 = rw \bmod n$
- 3 Compute  $P = u_1 \cdot G + u_2 \cdot Q$
- 4 Valid if  $x_P \equiv r \pmod{n}$

`charts/lesson_15/ecdsa_process.pdf`

## Wallet Components:

- Private key (spend authority)
- Public key (derived from private)
- Address (hash of public key)

## Key Derivation:

Private Key  $\xrightarrow{\text{ECC}}$  Public Key  $\xrightarrow{\text{Hash}}$  Address

**One-way:** Cannot derive private from address

[charts/lesson\\_15/wallet\\_key\\_hierarchy.pdf](#)

## Bitcoin Address Generation Pipeline



### Steps:

- 1 Generate 256-bit private key (random number)
- 2 Compute public key:  $\text{PubKey} = \text{PrivKey} \times G$  (secp256k1)
- 3 Hash the public key: SHA256 then RIPEMD160

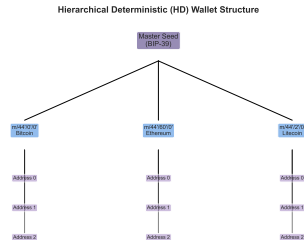
# Hierarchical Deterministic (HD) Wallets

## Problem:

- Managing multiple random keys
- Backup complexity
- Privacy (address reuse)

## Solution (BIP-32/44):

- Master seed (12/24 words)
- Derive infinite keys deterministically
- Hierarchical tree structure
- One backup for all keys



## Mnemonic Seed Phrases (BIP-39)

`charts/lesson_15/mnemonic_generation.pdf`



## Wallet Types: Hot vs Cold

`charts/lesson_15/wallet_types_comparison.pdf`

## Never Share:

- Private keys
- Seed phrases
- Wallet files without encryption

## Recommendations:

- Use hardware wallets for large amounts (Ledger, Trezor)
- Backup seed phrase offline (metal, paper in safe)
- Multi-signature for institutional custody
- Never store keys on cloud services
- Verify addresses carefully (malware can swap addresses)

**Warning:** Lost private key = Lost funds permanently

## Benefits:

- Self-custody (be your own bank)
- No intermediaries
- Censorship resistance
- Programmable ownership

## Challenges:

- User error (lost keys)
- No password reset
- Irreversible transactions
- Phishing attacks

[charts/lesson\\_15/key\\_security\\_threats.pdf](#)

- **Public Key Cryptography:** Two keys (public/private), trapdoor functions
- **ECC:** Efficient, smaller keys, basis for Bitcoin/Ethereum signatures
- **ECDSA:** Digital signatures prove transaction authorship
- **Wallets:** Manage private keys, addresses derived from public keys
- **HD Wallets:** One seed generates infinite keys (BIP-32/39/44)
- **Security:** Private key = ownership, loss is permanent

**Next Lesson:** Proof of Work – how cryptography secures the blockchain