

Unit 4 – Public-Key Cryptography

This unit gives an overview of asymmetric (or public-key) cryptography explaining the main concepts, key generation, encryption and decryption and its uses.

At the end of the unit, the learner should be able to:

- Explain the basic principles of public-key cryptosystems.
- Describe the uses of public-key cryptosystems.
- Describe the RSA, ECC, and DH algorithms

Target Audience: Third Year Students - BSc Software Engineering, BSc Computer Science

1. Introduction

"Public key cryptography (PKC), also known as asymmetric cryptography, revolutionizes secure communication by using mathematically linked key pairs: a widely distributable public key and a securely held private key. Unlike symmetric systems requiring shared secrets, PKC enables confidential message encryption (via the recipient's public key), digital signature generation/verification (using the sender's private/public keys), and secure key exchange protocols—all without pre-shared keys. This breakthrough solves critical key distribution challenges, establishes trust in open networks like the internet, and forms the foundation of modern security infrastructures including TLS/SSL, digital certificates, and blockchain systems."

Public-key cryptography (PKC) uses a **pair of** keys (in contrast with symmetric ciphers which use only one key) :

- **Public Key:** Shared openly; used for encryption or signature verification.
- **Private Key:** Kept secret; used for decryption or signing.

The core functions of PKC are as follows:

1. **Encryption/Decryption:**
 - Encrypt with the recipient's public key, decrypt with the recipient's private key.
2. **Digital Signatures:**
 - Sign with sender's private key, verify with sender's public key.
3. **Key Exchange:** Securely establish shared secrets between sender and recipient (e.g., Diffie-Hellman).

Two examples of commonly known public-key ciphers shall be examined, namely RSA and ECC.

2. RSA (Rivest-Shamir-Adleman)

The RSA public-key cryptographic system was devised by Ron Rivest, Adi Shamir & Len Adleman at MIT in 1977.

Algorithm Steps:

1. **Key Generation:**
 - Choose large primes p and q (e.g., 2048 or more bits).
 - Compute $n = p * q$ (modulus) and $\phi(n) = (p - 1)(q - 1)$.
 - Select e (public exponent) where $1 < e < \phi(n)$ and $\gcd(e, \phi(n)) = 1$.
 - Compute d (private exponent) such that $d * e \equiv 1 \mod \phi(n)$.
 - **Public Key:** (e, n)
 - **Private Key:** (d, n)
2. **Encryption:**
 - Ciphertext $c = m^e \mod n$ (where m is plaintext).
3. **Decryption:**
 - Plaintext $m = c^d \mod n$.

Security:

- Relies on **factoring large integers** (hard for classical computers).
- Vulnerable to **Shor's algorithm** on quantum computers.

Uses:

- TLS handshake
- Secure email (PGP)
- Digital signatures
- Secure software updates

3. ECC (Elliptic Curve Cryptography)

Concept:

- Based on algebraic structures of **elliptic curves** over finite fields.
- Curve equation: $y^2 = x^3 + ax + b$ (e.g., NIST curves: $P - 256$, curve *secp256k1* used in Bitcoin).

Key Generation:

- Private key d : Random integer.
- Public key Q : Point on curve, $Q = d * G$ (where G is a base point).

Operations:

- **Encryption/Decryption:** Uses ECC variants (e.g., ECIES).

- **Signatures:** ECDSA (Elliptic Curve Digital Signature Algorithm).

Security:

- Relies on **Elliptic Curve Discrete Logarithm Problem (ECDLP)**:
 - Given Q and G , finding d is computationally infeasible.
- Stronger per bit than RSA: **256-bit ECC $\approx$ 3072-bit RSA**.

4. Diffie-Hellman Key Exchange (DH)

Goal: Securely establish a shared secret over an insecure channel.

Steps (Classic Diffie-Hellman):

1. **Public Parameters:** Large prime p and generator g .
2. **Alice:**
 - Generates private key a , sends $A = g^a \bmod p$ to Bob.
3. **Bob:**
 - Generates private key b , sends $B = g^b \bmod p$ to Alice.
4. **Shared Secret:**
 - Alice computes $s = B^a \bmod p$.
 - Bob computes $s = A^b \bmod p$.
 - Result: $s = g^{a*b} \bmod p$.

5. Elliptic Curve Diffie-Hellman (ECDH):

- Uses elliptic curve points instead of modular arithmetic.
- Shared secret: $s = d_a * Q_b = d_b * Q_a$ (where Q is public key).

Security:

- Relies on **Discrete Logarithm Problem (DLP)** or **ECDLP**.
- Vulnerable to **man-in-the-middle attacks** (requires authentication via PKI).

6. Advantages & Disadvantages

Criteria	RSA	ECC
Key Size	Large (2048–4096 bits)	Small (256–384 bits)
Speed	Slower encryption/signing	Faster operations
Security	Weakened by quantum attacks	Resists quantum attacks better*
Maturity	Widely adopted, standardized	Growing adoption (NSA Suite B)
Use Cases	TLS, SSH, email encryption	IoT, mobile devices, blockchain

***Note:** ECC is quantum-vulnerable but requires larger key sizes than RSA for equivalent security.

7. Public Key Infrastructure (PKI)

Purpose: Bind public keys to identities (e.g., websites, users) via **digital certificates**.

Components:

1. **Certificate Authority (CA):**
 - Trusted entity issuing certificates (e.g., DigiCert, Let's Encrypt).
 - Signs certificates with its private key.
2. **Digital Certificate:**
 - Contains public key, owner info, validity period, and CA's signature.
 - Standards: X.509.
3. **Registration Authority (RA):** Verifies identities before CA issues certificates.
4. **Certificate Revocation:**
 - CRL (Certificate Revocation List) or OCSP (Online Certificate Status Protocol).

Role of RSA/ECC in PKI:

- **Certificates:** Store RSA/ECC public keys.
- **TLS Handshake:**
 - RSA/ECC used for key exchange (e.g., RSA key transport, ECDHE).
 - ECDSA/RSA for server authentication.
- **Document Signing:** Adobe PKI, code signing.

8. Summary & Best Practices

- **RSA:** Legacy systems, compatibility-focused environments.
- **ECC:** Resource-constrained systems (mobile/IoT), modern TLS.
- **Diffie-Hellman:** Secure key exchange (always use authenticated variants).
- **PKI:** Critical for trust; use strong keys (RSA $\geq$ 2048 bits, ECC $\geq$ 256 bits).

Future Trends:

- **Post-Quantum Cryptography (PQC):** NIST-standardized algorithms (e.g., CRYSTALS-Kyber).
- **Hybrid Schemes:** Combine RSA/ECC with PQC for transition.

9. Questions for students

1. What are the main differences between RSA and ECC in terms of security and efficiency?
2. Explain how Diffie-Hellman key exchange works and why it's important.
3. How does a PKI ensure that a public key truly belongs to a specific user or website?
4. Why is ECC preferred for mobile and embedded devices?
5. In what way does a digital certificate help prevent man-in-the-middle attacks?