

# UNIT 9

## Security

*Distributed Systems (CACS352) — BCA III/VI*

### ⚠ Please note

*These notes were prepared with AI assistance and **may contain errors or omissions**. Please cross-check definitions, algorithms and numerical examples against your prescribed textbook (e.g., Tanenbaum & van Steen, *Distributed Systems*), your class notes and your instructor before relying on them for exams.*

## 9.1 Introduction to Security

A distributed system is exposed to attacks over the network, so security must be designed in from the start. A secure system must provide the **CIA** properties: **Confidentiality** (information is disclosed only to authorised parties), **Integrity** (changes are made only in authorised ways), and **Availability** (services are accessible to authorised parties). Related goals are **authenticity**, **authorisation** and **non-repudiation**.

### Security threats

| Threat       | Meaning                                                               | Example                        |
|--------------|-----------------------------------------------------------------------|--------------------------------|
| Interception | Unauthorised party gains access to data or service                    | Eavesdropping, wiretapping     |
| Interruption | A service or data becomes unavailable or unusable                     | Denial of service (DoS)        |
| Modification | Unauthorised tampering with data or service                           | Altering a transaction amount  |
| Fabrication  | Generating additional data or activity that would not otherwise exist | Replay attack, forged messages |

A **security policy** states what actions entities may take; **security mechanisms** enforce it: **encryption**, **authentication**, **authorisation** and **auditing**.

### Design issues

- **Focus of control** — protect data, operations, or users (roles).
- **Layering of security mechanisms** — at application, middleware, transport or network level; the placement determines what must be trusted.

- **Distribution of security mechanisms** — the **Trusted Computing Base (TCB)** is the set of components that must work correctly for security; keep it **small** (simplicity — complex systems are less secure).
- Security by obscurity is unreliable — assume attackers know the algorithms; only keys are secret.

## Cryptography — a quick review

| Type                    | Keys                                                         | Examples  | Use                               |
|-------------------------|--------------------------------------------------------------|-----------|-----------------------------------|
| Symmetric               | Same secret key for encryption and decryption                | AES, 3DES | Bulk data encryption (fast)       |
| Asymmetric (public key) | Public key encrypts / verifies; private key decrypts / signs | RSA, ECC  | Key exchange, signatures          |
| Hash function           | No key; fixed-size digest, one-way, collision resistant      | SHA-256   | Integrity, MAC (HMAC), signatures |

### Example 9.1 — RSA with small numbers

Choose primes  $p = 3$ ,  $q = 11$ . Choose  $e = 7$ . Compute the keys and encrypt then decrypt the message  $M = 4$ .

#### Solution

$n = p \times q = 33$ ;  $\varphi(n) = (p - 1)(q - 1) = 2 \times 10 = 20$ .

$e = 7$  is coprime to 20. Find  $d$  with  $e \cdot d \equiv 1 \pmod{20}$ :  $7 \times 3 = 21 \equiv 1 \rightarrow \mathbf{d = 3}$ . Public key  $(7, 33)$ ; private key  $(3, 33)$ .

Encrypt:  $C = M^e \bmod n = 4^7 \bmod 33 = 16384 \bmod 33 = \mathbf{16}$ .

Decrypt:  $M = C^d \bmod n = 16^3 \bmod 33 = 4096 \bmod 33 = \mathbf{4} \checkmark$ .

Real RSA uses primes of 1024 bits or more.

## 9.2 Secure Channels

A **secure channel** protects senders and receivers against interception, modification and fabrication. It provides **mutual authentication** and **message confidentiality and integrity**.

## Authentication

- **Shared secret key (challenge–response)** — A sends its identity; B replies with a random challenge R; A returns R encrypted with the shared key K, proving knowledge of the key, and challenges B in the same way. Vulnerable to **reflection attacks** if the protocol is naive.
- **Key Distribution Centre (KDC)** — with N parties, pairwise keys need  $N(N - 1)/2$  keys; a KDC shares one key with each party and hands out session keys. The **Needham–Schroeder** protocol uses nonces and tickets; **Kerberos** is a practical KDC-based system.
- **Public-key authentication** — A sends a challenge encrypted with B's public key; only B can decrypt and respond. Requires trustworthy public keys (certificates, Section 9.5).

### Example 9.2 — Number of keys

In a system of 100 users, how many secret keys are needed if every pair has its own key? How many with a KDC?

### Solution

Pairwise:  $N(N - 1)/2 = 100 \times 99 / 2 = 4950$  keys. With a KDC: each user shares one key with the KDC  $\rightarrow$  **100** keys.

## Kerberos in brief

The user logs in and the **Authentication Server (AS)** gives a **Ticket-Granting Ticket (TGT)**. To use a service, the client shows the TGT to the **Ticket-Granting Server (TGS)** and receives a **service ticket** containing a session key; the client presents the ticket plus an authenticator (timestamp) to the service. Passwords are never sent, and tickets have limited lifetime; clocks must be loosely synchronised.

## Message integrity and confidentiality

Confidentiality uses encryption with a fresh **session key** (usually symmetric, agreed via key exchange). Integrity is checked with a **Message Authentication Code (MAC/HMAC)** or a **digital signature** (hash of the message signed with the sender's private key, verified with the public key; also gives non-repudiation). Replay is prevented with nonces, sequence numbers or timestamps.

## Diffie–Hellman key exchange

Two parties agree on public numbers  $p$  (prime) and  $g$ . A picks secret  $a$  and sends  $g^a \bmod p$ ; B picks secret  $b$  and sends  $g^b \bmod p$ . Both compute the shared key  $g^{ab} \bmod p$ . An eavesdropper sees only  $g^a$  and  $g^b$  and must solve the discrete logarithm problem. Without authentication it is vulnerable to **man-in-the-middle** attacks.

### Example 9.3 — Diffie–Hellman

$p = 23$ ,  $g = 5$ , Alice's secret  $a = 6$ , Bob's secret  $b = 15$ . Compute the shared key.

### Solution

Alice sends  $A = 5^6 \bmod 23 = 15625 \bmod 23 = 8$ . Bob sends  $B = 5^{15} \bmod 23 = 19$ .

Alice:  $B^a = 19^6 \bmod 23 = 2$ . Bob:  $A^b = 8^{15} \bmod 23 = 2$ . Shared key = 2.

## Secure channel in practice — TLS

**Transport Layer Security (TLS)** secures HTTP (HTTPS), e-mail, and many other protocols. The **handshake** negotiates parameters, authenticates the server with an X.509 certificate (optionally the client too) and derives session keys; the **record protocol** then encrypts and authenticates application data. Modern TLS 1.3 uses ephemeral Diffie–Hellman for **forward secrecy** and removes weak algorithms.

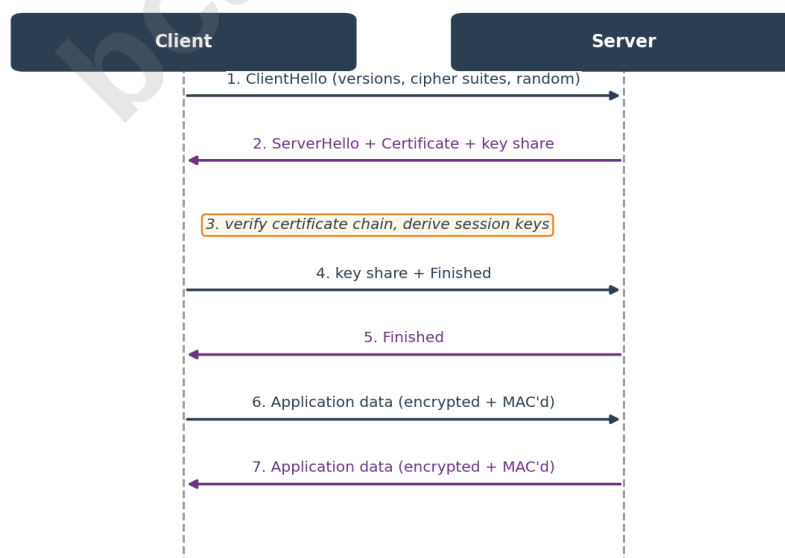


Figure 9.1 — Simplified TLS handshake and data exchange.

**Secure group communication** needs a shared group key: all members share one secret key (efficient but revocation requires re-keying), or each pair of members uses a separate key (secure but expensive).

## 9.3 Access Control

Authentication tells *who* you are; **authorisation** decides *what you may do*. An **access control** mechanism has a **subject** (user/process) invoking an **operation** on an **object**; a **reference monitor** intercepts every request and checks the policy.

### Access control matrix, ACLs and capabilities

Conceptually the rights are a matrix with subjects as rows and objects as columns. Since it is sparse, it is stored by column (**Access Control List, ACL**, kept with each object) or by row (**capability list**, kept with each subject).

| Subject \ Object | File A      | File B      | Printer |
|------------------|-------------|-------------|---------|
| Alice            | read, write | read        | print   |
| Bob              | read        | —           | print   |
| Carol            | —           | read, write | —       |

- **ACL** — for File A: {Alice: read, write; Bob: read}. Easy to review/revoke rights for an object.
- **Capability** — Alice holds {File A: rw, File B: r, Printer: print}. Easy delegation, but must be unforgeable (protected by the OS or by cryptography) and revocation is harder.

### Access control models

- **DAC** (discretionary) — owners decide who may access their objects (Unix permissions).
- **MAC** (mandatory) — a central policy with security labels (e.g., Secret/Top-Secret) decides.
- **RBAC** (role-based) — permissions are assigned to roles, users to roles (e.g., *teacher*, *student*, *admin*).

### Firewalls

A **firewall** separates an internal network from the Internet and checks all traffic. A **packet-filtering gateway** decides based on header fields (addresses, ports, protocol). An **application-level gateway (proxy)** inspects application data (e.g., an e-mail filter). Firewalls are not sufficient against insider attacks, malicious content in allowed traffic, or DoS.

## Secure mobile code

Downloaded code (JavaScript, applets, agents) may harm the host. Defences: the **sandbox** (restrict what the code may do, e.g., no local file access), a **playground** (separate isolated machine), **code signing** (verify the publisher, not the safety), and **resource limits**. For mobile agents the agent itself must also be protected from a malicious host.

## Denial of service

DoS attacks make a service unavailable by exhausting resources; in a **DDoS** attack thousands of compromised machines (botnet) participate. Defences: rate limiting, filtering, over-provisioning and CDNs, and detecting anomalies close to the source.

## 9.4 Secure Naming

Name services are attractive targets: if an attacker can make a name resolve to a wrong address, the client is directed to a fake server (**pharming**). **DNS cache poisoning** inserts bogus records into a resolver's cache. Secure naming aims to make sure that (1) the answer really comes from the authoritative source and (2) it has not been changed.

- **DNSSEC** — DNS records are **digitally signed** (RRSIG records) by the zone's key (DNSKEY); a parent zone vouches for the child's key with a DS record, forming a **chain of trust** from the root. Resolvers verify signatures. It gives authenticity and integrity, not confidentiality, and adds complexity (key rollover, larger responses).
- **Secure flat names / self-certifying names** — the identifier itself is derived from a public key:  $\text{id} = \text{hash}(\text{public key})$ . Anyone can check that a claimed owner holds the private key, so no third party is needed (e.g., self-certifying file names, onion addresses).
- **Human-friendly names** bound to keys through **certificates** issued by trusted authorities; **secure DHT lookups** must protect against nodes that lie in routing (e.g., Sybil attacks), using redundant routes and node id constraints.

## 9.5 Security Management

### Key management

- **Key establishment** — session keys are created by Diffie–Hellman or transported encrypted under a long-term key.
- **Key distribution** — symmetric keys via a KDC; public keys via **certificates**.
- **Key lifetime** — the more a key is used, the more material an attacker has; use short-lived session keys and replace long-term keys periodically.

- **Public-key certificate (X.509)** — binds an identity to a public key and is signed by a **Certification Authority (CA)**. Verification follows a **certificate chain** up to a trusted root. Together with CAs, registration and revocation this is called **Public Key Infrastructure (PKI)**.
- **Revocation** — a compromised certificate is listed on a **Certificate Revocation List (CRL)** or checked online with **OCSP**.

## Secure group management

Groups (e.g., secure multicast) need controlled admission, member removal and re-keying: a new member must not read past messages (backward secrecy) and a departed member must not read future ones (forward secrecy). Trees of keys (logical key hierarchy) reduce re-keying cost to  $O(\log N)$ .

## Authorisation management

- **Capabilities and attribute certificates** — signed statements that grant specific rights, so the server need not consult a central ACL.
- **Delegation** — a user lets another process act on their behalf with limited rights, using a **proxy** (a restricted certificate plus a private key) so that full credentials are not passed on. OAuth tokens are a modern example.
- **Least privilege** — give each principal only the rights it needs, and audit what happens.

## Trust and auditing

Systems keep **audit logs** of security-relevant events, review them, and use intrusion detection to spot attacks. Overall security also depends on people and processes — social engineering and weak passwords are as dangerous as protocol flaws.

### Key Takeaways

- ✓ Security goals: confidentiality, integrity, availability; threats: interception, interruption, modification, fabrication.
- ✓ Cryptography: symmetric (AES) for data, asymmetric (RSA, DH) for key exchange and signatures, hashes for integrity.
- ✓ Secure channels combine authentication, key exchange, encryption and MACs — TLS and Kerberos are the standard examples.
- ✓ Access control uses ACLs or capabilities under DAC, MAC or RBAC; firewalls, sandboxes and DoS defences protect the perimeter.

- ✓ Secure naming: DNSSEC chain of trust and self-certifying names.
- ✓ Security management: PKI/certificates, key lifetimes, group re-keying, delegation and auditing.

— *End of Unit 9* —

bcalink.com