Cybersecurity

Alessandro Dori

Contents

1 Fundamental Concepts

1.1 Computer Security Definitions

According to the National Institute of Standards and Technology (NIST), **computer security** consists of the prevention of damage, protection, and restoration of computers, electronic communications systems and services, and any other type of digital structure.
In this course, we define **computer security** as measures and controls that ensure the confidentiality, integrity, and availability (CIA) of information system assets (hardware, software, and information) being processed, stored, and communicated.

1.2 Essential Definitions

- **Asset**: Any resource, information, or entity related to the system.
- **Threat**: Any circumstance or event with the potential to adversely impact organizational operations.
- **Threat agent (Adversary)**: Anyone who conducts or intends to conduct detrimental activities.
- **Countermeasures**: A device or technique whose objective is to impair detrimental activities.
- **Risk**: A measure of the extent to which an asset is exposed to a threat. It includes the impact that would arise if an event occurs and the likelihood of that event.
- **Vulnerability**: A weakness in an information system, internal controls, implementation, etc., that could be exploited or triggered by a threat source.

Observation: The security of a system, application, or protocol is always relative to the set of desired properties and the capabilities of a potential threat agent.
Example: Standard file access permissions in Linux or Windows are ineffective against an adversary who can boot the system from an external device (e.g., CD or USB).

1.3 Types of Attacks

- **Active attack**: An attempt to alter system resources or affect their operation. Four categories of active attacks include:
  - Replay
  - Masquerade
  - Modification of messages
  - Denial of service
- **Passive attack**: An attempt to learn or make use of information from the system without affecting system resources. Two main categories:

    - Release of message contents
    - Traffic analysis

- **Inside attack**: Initiated by an entity inside the system's security perimeter (an "insider"), misusing authorized resources.
- **Outside attack**: Initiated by an entity outside the system's security perimeter.

1.4 Confidentiality, Integrity, and Availability (CIA)

1.4.1 Confidentiality

Confidentiality is the avoidance of the unauthorized disclosure of information. It implies protecting data so that only those who are allowed to see it can do so, while preventing everyone else from learning anything about its content.
Main Tools to Ensure Confidentiality:

- **Encryption**: Transforming information using a secret (encryption key) to make it readable only to those who know the decryption key.
- **Access control**: Defining rules and policies that limit access to confidential information to legitimate people or systems.
- **Authentication**: Determining the identity or role of an entity, based on factors like something the person has, knows, or is.
- **Authorization**: Determining if a person or system is allowed to access resources according to a policy.
- **Physical security**: Physical barriers to limit access to protected computational resources.

1.4.2 Integrity

Integrity is the property that something must not be altered in an unauthorized way. It often involves the use of backups, checksums, data-correcting codes, etc., to ensure data has not been tampered with.

1.4.3 Availability

Availability is the property that something is accessible and modifiable in a timely manner by authorized individuals. Physical protections and computational redundancies are typically used to ensure systems remain available.
These three concepts (confidentiality, integrity, and availability) form the **CIA Triad**, which should be preserved to ensure system security.

1.4.4 Other Security Concepts

- **Authenticity**: The ability to determine that statements, policies, and permissions issued by a person are genuine (including non-repudiation).
- **Accountability**: The requirement that an entity's actions be traceable uniquely to that entity (through activity records).
- **Anonymity**: The property that certain records or transactions are not attributable to any specific individual.

1.5 Threat Consequences and Types

1.5.1 Categorization by CIA Impact

- **Unauthorized disclosure**: An entity gains access to data for which it is not authorized (threat to confidentiality).
- **Deception**: An authorized entity receives false data and believes it to be true (threat to system or data integrity).
- **Disruption**: Interrupts or prevents correct operation of system services (threat to availability or system integrity).
- **Usurpation**: Results in control of system services or functions by an unauthorized entity (threat to system integrity).

1.5.2 Categorization by Attack Type

- **Interception**: Eavesdropping on information during its transmission.
- **Falsification**: Unauthorized modification of information (e.g., man-in-the-middle attacks).
- **Denial of service (DoS)**: Obstruction or degradation of data service or information access.
- **Masquerading**: Fabrication of information claiming to be from someone else.
- **Repudiation**: Denial of commitment or data reception (e.g., attempting to back out of a contract or protocol).
- **Inference (correlation/traceback)**: Combining multiple data sources/flows to deduce a data stream's source or content.

1.6 Authentication

1.6.1 General Concepts

Authentication is the process of establishing confidence in the user identities presented to an information system electronically. Typically, it relies on one or more of the following:

- **Something you know**: e.g., a password or PIN.
- **Something you possess**: e.g., a token or key card.
- **Something you are**: e.g., biometrics (fingerprints, iris, face).
- **Something you do**: e.g., dynamic biometrics (handwriting, voice pattern).

Using multiple authentication methods (multifactor authentication) improves security.

1.6.2 Passwords

Passwords are one of the most common means of authentication. The user provides a username and password, compared by the system to stored credentials.
Hash Functions: A hash function is a one-way function (irreversible) that converts a string of text into a fixed-length, incomprehensible *hash*.
Storing passwords as hashes is more secure. A good hash function should be:

- Efficient to compute.
- Able to minimize collisions (two different inputs generating the same output).

Salt: A salt is a fixed-length string appended to the original input before hashing. This makes the output more secure by increasing the input size.
Example: Modern UNIX systems store passwords by looping the MD5 hash 1000 times with a salt of up to 48 bits, producing a 128-bit hash value.

1.6.3 Password Attacks

- **Dictionary attacks**: Using a large list of possible passwords, hashing them with different salts, and comparing to the stored hash values.
- **Rainbow table attacks**: Using precomputed tables of hash values for all possible salts. Countermeasures include sufficiently large salts and large hash lengths.

A long enough salt renders these attacks computationally infeasible.

1.6.4 Tokens

A token is a small string of text used to identify a user or entity. Common examples include:

- **Magnetic stripe cards**: Plastic cards with a magnetic stripe containing personalized information; vulnerable to cloning attacks.
- **Smart tokens**: Small devices with embedded microprocessors (e.g., bank cards).
- **Smart cards**: Plastic cards containing a processor, memory, and small I/O ports (e.g., eID).

Less common examples involve **biometrics**. The system compares the user's measured biometric data (converted to a reference vector) against a stored template.
Biometric authentication systems can involve:

- **Enrollment**: User registers biometric data with a PIN.
- **Verification**: System matches the user's provided biometric data (and PIN) with a stored template.
- **Identification**: System identifies a user only by biometric data, searching for matches among stored templates.

A core challenge is the **biometric accuracy dilemma**: Because perfect matches are impossible, recognition relies on probability density functions.

1.6.5 Remote User Authentication

Modern systems allow remote user authentication over networks (e.g., the Internet). While convenient, these methods must address threats like eavesdropping and replay attacks. **Challenge-response protocols** are often used to mitigate these risks.

1.7 Access Control

1.7.1 Definition of Access Control

Access control is the process by which use of system resources is regulated according to a security policy, allowing only authorized entities to access those resources.

1.7.2 Discretionary Access Control (DAC)

In **DAC**, access is controlled based on the identity of the requestor and rules stating what requestors are allowed to do. An entity with access rights can enable other entities to access that resource.
Common DAC Implementations:

- **Access Control Matrix**: A matrix with subjects (users) on one dimension and objects (resources) on the other; each entry specifies the subject's rights on the object.

- **Extended Access Control Matrix**: Includes the ability for a subject to create other subjects and have "owner" rights to them.
- **Access Control List (ACL)**: Each object has a list of subjects and their respective rights.
- **UNIX File Access Control**: A simplified ACL with owner (User ID), group (Group ID), and 12 protection bits (read, write, execute for owner, group, others).
- **Capability List**: Each subject has a list of objects and the rights granted on each object.
- **Access Control Function**: Every access is mediated by a controller that checks an up-to-date matrix or list to see if the subject has the correct permissions.

1.7.3 Mandatory Access Control (MAC)

In **MAC**, each subject and object is assigned a security class/level (forming a strict hierarchy). A subject is said to have a *security clearance*, and an object has a *security classification*. Together with **Multilevel Security (MLS)**, we have four access modes:

- read, append, write, execute

Confidentiality in MAC requires:

- No read up: A subject cannot read objects at a higher level.
- No write down: A subject cannot write to objects at a lower level.

1.7.4 Role-based Access Control (RBAC)

In **RBAC**, access rights are tied to roles rather than directly to subjects. Users are assigned to roles, and roles are assigned permissions. This is more flexible and scalable.
Example:

- **User assignment table** associates users to roles.
- **Permission assignment table** associates permissions to roles.

RBAC Sub-models:

- **RBAC0**: The basic model.
- **RBAC1**: Adds role hierarchies (sub-roles inherit permissions from super-roles).
- **RBAC2**: Replaces hierarchies with constraints (e.g., mutually exclusive roles, cardinality, prerequisite roles).
- **RBAC3**: Combines role hierarchies and constraints.

1.7.5 Attribute-based Access Control (ABAC)

In **ABAC**, attributes of both resources and subjects are used to define permissions, allowing finer-grained policies without an explosion of roles. There are three types of attributes:

- **Subject attributes**: Identity or characteristics of the subject.
- **Object attributes**: Properties of the resource (object).
- **Environmental attributes**: External context (time, location, system status).

Policies: A set of rules and relationships that govern allowable behavior within an organization.
Example: Restricting access to movies based on the user's age and the movie's rating. An ABAC policy can define conditions involving user attributes (e.g., age) and object attributes (e.g., movie rating).

Comparison with RBAC. While RBAC might require many roles as attributes grow, ABAC can more flexibly handle large numbers of attributes without role explosion.

2 Vulnerabilities and Countermeasures

2.1 Malware

Definition 1 (Malware)**.** *Any program inserted into a system with the intent of compromising the confidentiality, integrity, or availability of the victim's data, applications, or operating system or otherwise annoying or disrupting the victim.*

Malware is typically classified based on:

- **Propagation mechanism**: How the malware spreads (e.g., virus-infected files, worms exploiting vulnerabilities, Trojans, and phishing-based social engineering).
- **Payload actions**: The malicious actions (e.g., corrupting data files, turning the system into a "zombie" bot, keylogging, hiding its presence).

Evolution of Malware

- Initially required significant technical skill to develop.
- Emergence of virus-creation toolkits and crimeware made malware deployment accessible even to non-experts.
- Motivation shifted from individual technical prowess to organized, well-funded attackers (criminal enterprises, state-sponsored organizations).

Definition 2 (Advanced Persistent Threat (APT))**.** *A well-resourced and persistent application of a wide variety of intrusion technologies and malware against carefully selected targets (often attributed to state-sponsored or highly organized entities).*

Characteristics of APTs:

- **Advanced**: Use a wide variety of intrusion technologies and custom malware tailored to the target.
- **Persistent**: Determined attacks over an extended period, applying multiple attack methods progressively until compromise is achieved.
- **Threat**: Organized, capable, and well-funded attackers specifically targeting selected entities, greatly raising the likelihood of success.

2.1.1 Types of Malware

Virus A **virus** is a piece of software that infects other programs by modifying them to include a copy of itself, enabling it to replicate and spread. Main components:

- **Infection mechanism (infection vector)**: How the virus propagates.
- **Trigger (logic bomb)**: Condition(s) determining when the payload executes.
- **Payload**: The destructive or harmful actions (or harmless pranks).

Virus Life Cycle:

1. **Dormant phase**: Virus is idle, awaiting activation.
2. **Triggering phase**: Virus is activated by some event or condition.
3. **Propagation phase**: Virus replicates itself by infecting other programs or system areas.
4. **Execution phase**: The virus performs its malicious payload.

Macro Viruses Viruses that infect documents with macro capabilities (e.g., Microsoft Word). They:

- Are platform independent.
- Use embedded scripting in documents.
- Are easy to write and spread rapidly.

Virus Classification

- **By target**:
  - *Boot sector infector*
  - *File infector*
  - *Macro virus*

  - *Multipartite virus*

- **By concealment strategy**:
  - *Encrypted virus*: Uses a random key to encrypt part of itself.
  - *Stealth virus*: Explicitly designed to hide from anti-virus software.
  - *Polymorphic virus*: Mutates with every infection.
  - *Metamorphic virus*: Rewrites itself completely at each iteration, possibly changing both appearance and behavior.

Worm A **worm** is a program that actively seeks out more machines to infect; each compromised system can then launch attacks on other machines. Worms:

- Replicate themselves without modifying host files.
- Often can be *remotely* controlled (unlike viruses, which generally run independently after infection).
- Exploit software vulnerabilities or use social engineering, and generally carry a payload.

Observation 1 (Viruses vs. Worms)**.** *A worm does not modify the host program; it replicates to cause slowdowns or is controlled remotely, while viruses insert themselves into other programs.*

Worm Replication Possible mechanisms:

- E-mail/IM attachments
- File sharing
- Remote execution
- Remote file access/transfer
- Remote login

Target Discovery Methods

- **Scanning** (random, hit-list, topological, local subnet).
- **Drive-by-Downloads**, **Watering-Hole Attacks**, and **Malvertising** are other (often web-based) techniques that install malware on victim machines without user knowledge.

Clickjacking Tricking a user into clicking on something different from what they perceive, e.g., hiding malicious functionality behind a legitimate button, or hijacking keystrokes.

Ransomware Malware that encrypts user files and demands payment for decryption. Attackers may threaten to publish or permanently destroy data if payment is not made.

2.1.2 Malware Payload Types

- **System corruption**: Causing damage or rendering hardware/software inoperable (e.g., logic bombs).
- **Attack agent bots**: Infected systems become part of a botnet controlled by a remote C&C (Command & Control) facility (IRC servers, P2P protocols, etc.).
- **Information theft**:
  - *Keyloggers*: Capture keystrokes.
  - *Spyware*: Monitors user activities, intercepts requests, modifies data in transit, etc.
  - *Phishing*: Social engineering technique to masquerade as a trusted entity and steal user credentials.

    Spear-phishing: Targeted approach using personal information to make the attack more convincing.
- **Stealthing attacks**:
  - *Backdoor stealthing*: Secret entry point that bypasses standard access procedures.
  - *Rootkit*: A set of hidden programs that maintain covert access to a system by subverting OS security mechanisms.

2.1.3 Malware Countermeasure Approaches

Ideal Goal: Prevention via awareness, security policies, and threat mitigation. If prevention fails, use technical measures:

- **Antivirus software**:
  - **Generation I**: Simple scanners matching known signatures.
  - **Generation II**: Heuristic scanners searching for suspicious patterns.
  - **Generation III**: Activity traps monitoring actions in memory (behavior-based).
  - **Generation IV**: Combined techniques (full-featured protection).
- **Sandbox analysis**: Suspicious code is executed in a virtualized environment to observe malicious behavior. Attackers may delay payload execution to evade detection.
- **Host-based behavior blocking**: Monitors program actions in real time to block suspicious activities.
- **Perimeter scanning and filtering**: Integrated into e-mail/web proxies at the firewall.

2.2 Denial of Service (DoS)

Definition 3 (Denial of Service (DoS))**.** *Any action preventing or impairing authorized use of networks, systems, or applications by exhausting resources (CPU, memory, bandwidth, etc.).*

Effect on Availability

- Overload network bandwidth, system resources, or application resources.
- Flood the network handling software with many valid requests, each consuming significant resources.

Classic DoS Flooding Attacks

- **Ping flood** (ICMP).
- **UDP flood**.
- **TCP SYN flood**.

SYN spoofing: Overflows the connection queues by sending large numbers of spoofed SYN packets, preventing legitimate connections.

Definition 4 (Distributed DoS (DDoS))**.** *A DoS attack launched from a large number of compromised systems (botnets). The attacker coordinates agents (zombies) via handler systems.*

HTTP Flood Bots bombard a website with HTTP requests (e.g., recursively following all links), overloading the server. *Slowloris* specifically monopolizes server connections by sending partial requests very slowly.

Reflection & Amplification Attacks

- **Reflection**: Spoof the victim's address in requests to a service; the service "reflects" replies onto the victim.
- **Amplification**: Use protocols (e.g., DNS, memcached) that generate large responses from small requests. Attackers leverage this to amplify traffic directed to the victim.

Prevention is Hard because high traffic volumes may be legitimate. Typical countermeasures include:

- **Blocking spoofed source addresses**.
- **Filter-based route checks**.
- **Modified TCP handling**.
- **Mirrored/replicated servers** (for high-availability needs).

2.3 Buffer Overflow

Definition 5 (Buffer overflow)**.** *A condition under which more input is placed into a buffer than its allocated capacity, overwriting adjacent memory and potentially corrupting data or executing malicious code.*

Buffer overflows are common, especially in lower-level languages with manual memory management. They can:

- Overwrite adjacent data.
- Alter control flow by overwriting return addresses or function pointers.
- Lead to injection of arbitrary shellcode.

Stack Overflow Example A classic case in C code using `gets()`, where input longer than the destination array causes adjacent memory (including other variables) to be overwritten.

Definition 6 (Shellcode)**.** *Machine code (specific to the processor/OS) designed to be injected and executed, often used in buffer overflow exploits.*

Shellcode must be *position independent* to run anywhere in the memory. Commonly, the attacker attempts to overwrite the return address in a function's stack frame, causing execution to jump to the injected shellcode.

2.3.1 Buffer Overflow Countermeasures and Variants

Compile-Time Defenses:

- Use of *safe* modern languages with built-in checks (though more resource-intensive).
- *Safer libraries* (replacing old unsafe library functions).
- *Stack protection* (e.g., StackShield, Return Address Defender) to detect overwrites.

Run-Time Defenses:

- *Executable address space protection*: Mark memory areas (stack, heap) as non-executable.
- *Address space randomization*: Randomize the location of key data structures in each process.
- *Guard pages*: Insert pages that trigger exceptions on any illegal access between memory regions.

Common Variants of Buffer Overflows:

- *Replacement stack frame*: Overwrite return addresses and frame pointers.
- *Return to system call*: Overwrite return address to call standard library functions with chosen arguments.
- *Heap overflow*: Overwrites structures in the dynamic memory region.
- *Global data overflow*: Overwrites global variables or function pointers.

2.4 Database Security

Relational databases store data in tables (rows and columns). *SQL (Structured Query Language)* is used to define, manipulate, and query data. Popularity of SQL makes it a target for attacks, particularly **SQL injection**.

2.4.1 SQL Injection

Sends malicious SQL commands to the DB server, typically aiming to:

- *Extract data* without authorization.
- *Modify or delete data.*
- *Execute OS commands.*
- *Launch DoS attacks.*

Mechanism Usually involves prematurely terminating a string and appending extra commands, often ending with a comment mark (`-`) to ignore subsequent query fragments.

SQL Sinks Places in a web application where user input or HTTP headers directly feed into SQL queries (e.g., GET/POST parameters, cookies).

Inband Attacks

- *Tautology*: Force conditions to always be true.
- *End-of-line comment*: Append `-` to disregard subsequent query parts.
- *Union query*: Combine results from multiple queries (e.g., extracting data from the `users` table).
- *Piggybacked queries*: Append additional queries to the legitimate query.

Inferential Attacks

- *Illegal/logically incorrect queries*: Reveal structure and DB types via error messages.
- *Blind SQL injection*: Use conditional responses or time delays to infer data from the DB.
- *Out-of-band attack*: Use alternative channels if direct responses are restricted.
- *File operations*: Read/write files on the server if SQL engine privileges allow.

Prevention Techniques:

- *Input sanitization* (escaping special characters).
- *Parametrized queries* (prepared statements).
- *SQL DOM* (strongly typed schema-based queries).

2.4.2 Database Authorization, Inference, and Encryption

Database Access Control Defines which portions of the DB are accessible and with which rights. Policies include:

- Centralized administration
- Ownership-based administration
- Decentralized administration

`GRANT` and `REVOKE` commands have cascading effects.

Inference The process of deducing unauthorized information from legitimately authorized queries or responses. Mitigation can happen:

- During DB design (altering structure, changing access control).
- During query execution (detecting potential inference channels).

Database Encryption Can be applied at various granularities (whole DB, tables, columns, individual entries). Challenges:

- Key management (authorized users must decrypt).
- Inflexibility (indexing/searching on encrypted data is more complex).

Partial solutions involve encrypting rows in blocks and partitioning attribute ranges for queries.

2.5 Web Security

2.5.1 HTTP Security Measures

HTTP is stateless; sessions are maintained by using **cookies**. The server stores session data; the client stores a session ID cookie to identify its session in subsequent requests.

- **Vulnerabilities**: Session hijacking if an attacker obtains or predicts the ID.
- **Session fixation**: Attacker sends a valid session ID to the victim, causing them to use that ID unknowingly.

Definition 7 (Insecure Direct Object Reference (IDOR))**.** *A vulnerability where the application exposes references to objects (e.g., via numeric IDs in URLs) without proper authorization checks.*

Same Origin Policy (SOP) Two scripts can access each other's DOM only if they share the same protocol, domain, and port.
Prevents malicious sites from accessing data on other sites.

2.5.2 XSS and CSRF Attacks

Cross-Site Scripting (XSS) Injecting malicious scripts into a legitimate page or web application, typically through lack of input sanitization. Types:

- *Reflected XSS*: Code injected in a parameter reflected in the server's response.
- *Stored XSS*: Code stored on the server (e.g., in a forum post) and served to other users.
- *DOM-based XSS*: Injection happens in a script context within the page itself.

Cross-Site Request Forgery (CSRF) Exploits the website's trust in the user's browser. An authenticated user is tricked into sending a malicious request (e.g., transferring money, changing account settings). *Because the user is logged in, the request includes valid credentials/cookies.*
Key Difference:

- **XSS**: Exploits the user's trust in a website (malicious code runs in the user's browser).
- **CSRF**: Exploits the website's trust in the user's browser (session cookies).

3 Cryptography

Purpose: Cryptography is primarily used to alter a message so that only intended recipients can reverse the process and read the original message. It preserves confidentiality, authenticates senders and receivers, and facilitates message integrity.
Let:

- K: the secret key
- P: the plaintext message
- C: the ciphertext (encrypted message)
- $E_K(P) = C$: the encryption function under key K
- $D_K(C) = P$: the decryption function under key K
- Both encryption and decryption are permutation functions on the set of all n-bit arrays

Example If Alice wants to send a message P to Bob over an insecure channel, they can use a *symmetric* encryption scheme with secret key K:

$$\text{Alice sends } E_K(P) \quad \longrightarrow \quad \text{Bob computes } D_K(E_K(P)) = P.$$

Definition 8 (Exhaustive search)**.** *The process of testing* every possible key K *in order to decrypt a given ciphertext.*

Because of brute force attacks, secret keys must be sufficiently long and random. Smaller keys can be exhaustively searched in little time. Larger keys make the required time to search infeasibly large.

Cryptanalysis Techniques An attacker might have:

- *Ciphertext-only attack*: Only a collection of ciphertexts.
- *Known-plaintext attack*: Several plaintext-ciphertext pairs.
- *Chosen-plaintext attack*: Plaintext-ciphertext pairs for plaintexts chosen by the attacker.
- *Chosen-ciphertext attack*: Plaintext-ciphertext pairs for ciphertexts chosen by the attacker.

Definition 9 (Symmetric and Public key cryptography)**.** Symmetric key cryptography*: a single secret key is used for both encryption and decryption.*
Public key cryptography (asymmetric cryptography)*: uses a public key for encryption and a private key for decryption.*

3.1 Symmetric Key Cryptography

3.1.1 Substitution and Transposition Ciphers

Substitution Ciphers Each character in the plaintext is replaced by another character from (potentially) the same alphabet.
Method 1: Caesar cipher

- Each plaintext character is replaced by the character K positions ahead (cyclic).
- For $K = 3$, `A` $\to$ `D`, `B` $\to$ `E`, etc.
- Plaintext `HELLO WORLD!` $\to$ `KHOOR ZRUOG!`

Only 26 possible keys make brute force easy. More advanced random-permutation ciphers use a permutation of the alphabet as the key, e.g.:

$$\texttt{A B C D E F G ... Z} \quad \longrightarrow \quad \texttt{K E P A L M U ... C}.$$

Still, $26! \approx 4.03 \times 10^{26}$ is large but vulnerable to *frequency analysis*.
Poly-alphabetic ciphers (e.g., using multiple alphabets or multiple keys) are more difficult to cryptanalyze.

Vigenère Cipher (Method 2)

- A word K is chosen as the key.
- Each letter of K corresponds to its position in the alphabet (e.g., `E` $= 5$).
- Characters of P are paired with characters of K (repeated if needed).
- Each plaintext character is shifted according to the numeric value of the key character.

If K = `USE THIS` and P = `THIS IS MY SECRET TEXT`, each character in P pairs with a character in K. The resulting ciphertext might be `NZML PA ES KIVYML NWBM`.
One-time pad:

- Key is as long as the plaintext.
- *Provably* unbreakable (Shannon's theorem).
- Disadvantage: keys cannot be reused and must be as long as the message.

Definition 10 (Shannon's Theorem)**.** *If a cipher is perfect, there must be at least as many keys as there are possible messages*

Transposition Ciphers Instead of substituting characters, *transposition* ciphers reorder the plaintext characters (no change in character frequency). Examples:

- *Rail fence transposition*: the given message gets arranged in a zig-zag pattern and then read row by row
- *Block transposition*: the given message gets divided in blocks of the same length and each character in the block gets reordered with the same permutation (the key)
- *Column transposition*: the given message gets written in a tabular form using a fixed row length, which then gets read column by column, forming the ciphertext
- *Keyed column transposition*: same as column transposition, but the columns get reordered by a permutation

Observation 2 (Observation 4)**.** *A cipher is* computationally secure *if:*

1. *The cost to break it exceeds the value of the information.*
2. *The time required to break it exceeds the useful lifetime of the information.*

3.1.2 Block and Stream Ciphers

Feistel Network Proposed by Horst Feistel (1973), a *product cipher* that combines two or more simple ciphers (substitution/permutation) in multiple rounds, strengthening cryptographic security.

Block Ciphers

- Plaintext and ciphertext blocks have fixed length n (e.g., 128 bits).
- Plaintext is partitioned into m blocks. Padding may be added to the last block.
- Each block is encrypted independently using the same key.

Data Encryption Standard (DES)

- Most widely used historically.
- 56-bit key, 64-bit block size. The 56-bit key eventually proved too short.
- Based on a Feistel network (Data Encryption Algorithm).

Double DES and Meet-in-the-Middle Attack

- Double DES attempts to use two keys ($56 + 56$ bits).
- Meet-in-the-middle reduces complexity from 2^{112} to about 2^{57} by encrypting from one side and decrypting from the other, looking for a match.

Triple DES (3DES)

- Uses 3 keys (or 2 keys, depending on the variant) in sequence.
- Fixes the design flaw of Double DES.

Advanced Encryption Standard (AES)

- Based on the Rijndael algorithm.
- Block size of 128 bits, key sizes of 128, 192, or 256 bits.
- 10 (128-bit key), 12 (192-bit key), or 14 (256-bit key) rounds of substitution, permutation, matrix multiplication, and XOR with round keys.
- each round consists of four operations:
  - ***SubBytes*** substitution is applied, where a number is split in 2 block used as indexes of a substitution table
  - ***ShiftRows*** a permutation step
  - ***MixColumns*** a matrix multiplication step
  - ***AddRoundKey*** a XOR step with a round key derived from the main key

Stream Ciphers

- Treat the message as a *continuous stream* of characters/bits.
- Key or encryption scheme can change for each character.
- Useful for processing data character-by-character or short messages.

Advantages: speed, no error propagation. Disadvantages: low diffusion and possible malicious insertions.

RC4

- Widely used stream cipher.
- Key forms a random permutation of all 8-bit values.
- Proven secure against known attacks if used correctly.

3.1.3 Block Cipher Modes

Electronic Code Book (ECB) Is the simplest mode of operation where each block is encrypted independently.

- Plaintext block $P[i]$ gets encrypted into the ciphertext block $C[i] = E_K(P[i])$.
- Ciphertext block $C[i]$ gets decrypted into the plaintext block $P[i] = D_K(C[i])$.
- Easy parallelization; tolerates block loss.
- Not good for encrypting documents/images with repeating patterns (blocks encrypt identically).

Cipher Block Chaining (CBC)

- Each block is XORed with the previous ciphertext block before encryption.
- $C[i] = E_K\Big(P[i] \oplus C[i-1]\Big)$; $C[-1] = V$ is the IV (initialization vector).
- Decryption: $P[i] = D_K(C[i]) \oplus C[i-1]$.
- Hides plaintext patterns but requires sequential block processing (not suitable when packets can be lost).

Cipher Feed Back (CFB) The message is treated as a stream of bits and gets XORed with the output of the block cipher

- Stream-like encryption.
- $C[i] = P[i] \oplus E_K\Big(C[i-1]\Big)$; $C[-1] = V$.
- Decryption: $P[i] = C[i] \oplus E_K\Big(C[i-1]\Big)$.

Output Feed Back (OFB) The message is treated as a stream of bits and gets XORed with the output of the block cipher and the encryption is applied to the partial output values $O[0], ..., O[N]$.

- Also stream-oriented.
- Each partial output $O[i] = E_K\Big(O[i-1]\Big)$; $O[0] = V$.
- $C[i] = P[i] \oplus O[i]$ and $P[i] = C[i] \oplus O[i]$.
- Sender and receiver must stay synchronized; the same (K, IV) must never be reused.

Counter (CTR) is conceptually identical to OFB, but allows the encryption to be parallelized thanks to each partial values being the encryption of the current counter instead of the previous partial value.

- Similar to OFB, but $O[i] = E_K(i)$, where i is a counter.
- Allows parallel encryption/decryption.

3.2 Message Authentication

Definition 11 (Message authentication)**.** *The process of verifying the* source *of a message and its* integrity*.*

It defends against active attacks. Authentication can be separate from encryption. Encryption alone does not guarantee authentication; they can be combined by attaching an *authentication tag* to the encrypted message.

Message Authentication Code (MAC) Generated with a secret key and an algorithm (often a hash function). A MAC is:

- Not necessarily reversible.
- A fingerprint of the message combined with a secret key.

Hash Function Requirements:

1. Accepts input of any size.
2. Produces a fixed-length output.
3. Relatively easy to compute $H(x)$.
4. *One-way (pre-image resistant)*: Infeasible to find x given h where $H(x) = h$.
5. *Second pre-image resistant*: Infeasible to find $y \neq x$ with $H(y) = H(x)$.
6. *Collision resistant*: Infeasible to find *any* pair (x, y) with $H(x) = H(y)$.

Observation 3 (Observation 5: Second pre-image vs. Collision)**.** *Second pre-image resistance is tested against a* given *block x, while collision resistance requires finding* any *two distinct blocks.*

Observation 4 (Observation 6)**.** *The strength of hash functions depends on the length of the hash code. Sufficiently long hashes ensure the previous properties hold.*

SHA (Secure Hash Algorithm)

- SHA-1 (160-bit), broken by *birthday attacks* with complexity $\approx 2^{80}$.
- SHA-2 (256–512 bits). Same structure as SHA-1, but larger outputs.
- SHA-3 designed in 2015 to replace SHA-2 if/when vulnerabilities are found.

HMAC (Hash-based MAC)

- A standard MAC mechanism for IP-layer security.
- Based on available hash functions (easily replaced if needed).
- Protects message integrity and authenticity.

$$\text{HMAC}(K, M) = H\Big[(K^+ \oplus \text{opad}) \,\|\, H\big[(K^+ \oplus \text{ipad}) \,\|\, M\big]\Big].$$

Security depends on the underlying hash function.

3.3 Public Key Cryptography

Overview Each user generates a *public* and *private* key. The public key is published.

- **Confidentiality:** A message encrypted with the *recipient*'s public key can only be decrypted by the corresponding private key.
- **Authentication:** A message "encrypted" (signed) with the *sender*'s private key can be verified with the sender's public key.

Public key cryptography is computationally expensive. Keys must be verified to ensure they're valid and not tampered with.

Definition 12 (Digital signature)**.** *A signature acknowledging content by encrypting a message digest of the content using the owner's private key.*

Using PKC

- *Confidentiality*: Encrypt with recipient's public key; decrypt with recipient's private key.
- *Authentication*: Encrypt with sender's private key; decrypt with sender's public key.
- *Message integrity*: Attach (and verify) a message digest.
- *Authentication + Integrity*: Achieved via *digital signatures*.

Hybrid Cryptography and Key Exchange Often combine symmetric and asymmetric cryptography:

- Use PKC to securely exchange a symmetric key.
- Then use the symmetric key for bulk encryption.

Vulnerable to *man-in-the-middle* attacks if public keys aren't authenticated.

3.3.1 Digital Certificates and PKI

Definition 13 (Digital certificate). *A document certifying the relationship between a public key and its owner, via a digital signature from a trusted Certification Authority (CA).*

Certification Authority (CA) responsibilities:

- Verify identities and issue key pairs.
- Maintain a register of valid keys.
- Revoke keys if they expire or are compromised (Certificate Revocation List, CRL).

Prevents man-in-the-middle attacks. If the CA is compromised, all stored keys are at risk. *Public Key Infrastructure (PKI)* organizes CAs in a hierarchy up to the root CA.
X.509 Standard

- Widely used format for public-key certificates.
- Contains public key, owner identity, and CA signature.
- Variants:
  - **Long-lived certificates**: where the CA and the end user are certificated. Typically issued for validity periods of months to years.
  - **Short-lived certificates**: where the user provides authentication for applications such as grid computing, while avoiding some of the overheads and limitations of conventional certificates. They have validity periods of hours to days, which limits the period of misuse if the certificate is compromised.
  - Other variants include **attribute certificates** and **proxy certificates**.

3.3.2 Public Key Algorithms

RSA Algorithm Named after Rivest, Shamir, and Adleman. Security relies on the difficulty of factoring the product of two large primes.

1. Choose large primes p, q and compute $n = pq$.
2. Compute $\varphi(n) = (p-1)(q-1)$.
3. Choose e with $\gcd(e, \varphi(n)) = 1$.
4. Find d such that $de \equiv 1 \pmod{\varphi(n)}$.
5. Public key: (n, e); Private key: (n, d).
6. Encryption: $C \equiv M^e \pmod{n}$.
7. Decryption: $M \equiv C^d \pmod{n}$.

Large p, q are required to prevent factoring. Timing attacks observe decryption times to infer d. Countermeasures: constant exponentiation time, random delay, or blinding.

Diffie-Hellman Key Exchange

- First published public-key algorithm, but used *only* for key exchange.
- Security is based on the difficulty of computing discrete logarithms modulo a large prime.
- Vulnerable to man-in-the-middle attacks if no authentication is provided.

Other Algorithms

- **Digital Signature Standard (DSS)**: Uses SHA-1 and DSA; only for signatures.
- **Elliptic-Curve Cryptography (ECC)**: Security equal to RSA with smaller key sizes.
- **El Gamal**: Based on Diffie-Hellman key exchange.

4 Internet Security Protocols and Standards

4.1 Email Security Protocols

MIME & S/MIME

- **MIME (Multipurpose Internet Mail Extension)**: Extends RFC822 by defining new header fields that describe message body content. No built-in security.
- **S/MIME**: Provides the ability to sign and/or encrypt e-mail messages via RSA or similar technology.

S/MIME supports:

- *Enveloped data*: encrypted content and associated keys.
- *Signed data*: ciphertext and signed digest.
- *Clear-signed data*: plaintext and signed digest.
- *Signed & enveloped data*: nesting signed and encrypted entities.

Signature + Encryption Example in S/MIME:

1. *Signing (RSA/DSA + SHA-256)*:
   - Compute a 256-bit digest (SHA-256).
   - Encrypt the digest with the sender's private RSA key (digital signature).
   - Append this signature to the message.
2. *Encryption (AES + RSA)*:
   - Generate a pseudorandom secret key (for AES) for each message.

  - Encrypt the message with this secret key.
  - Encrypt the secret key with the recipient's *public* RSA key.
  - Recipient uses her *private* RSA key to recover the secret key, then decrypts the message using AES.

DKIM (DomainKeys Identified Mail)

- Provides a cryptographic signature in the message header.
- *Signing domain* claims responsibility for the message.
- Signature is verified using the domain's *public key* fetched via DNS.
- Transparent to the end user: implemented on the Mail Delivery Agent, not the MUA.
- Prevents forgeries/masquerade attacks by verifying the sending domain.

Comparison: S/MIME vs. DKIM

- **S/MIME**:
  - End-to-end security; relies on sender & recipient having S/MIME-capable clients.
  - Signs the *message content*, but not necessarily the full header.
- **DKIM**:
  - Transparent to users (implemented on sending/receiving mail servers).
  - Authenticates the *domain* that originated the message.

4.2 Transport Layer Security (TLS)

SSL and TLS

- SSL was developed by Netscape; later standardized as TLS (RFC4336).
- Rely on TCP, providing a general secure service for applications (e.g., HTTPS).
- Two main concepts:
  - *TLS Session*: An association between client and server created by the Handshake protocol (defines security parameters).
  - *TLS Connection*: A transport protocol with one session.

Definition 14 (Cipher suite)**.** *A set of cryptographic algorithms (key exchange, encryption, MAC) agreed upon by two peers for a secure connection.*

TLS Record Protocol Used by four specific TLS protocols:

- **Change Cipher Spec Protocol**: A single byte with value 1. Its purpose is to update the *pending* state to the *current* state (i.e., switch to a newly negotiated cipher suite).
- **Alert Protocol**: Conveys TLS-related alerts (warning or fatal). Fatal alerts terminate the connection immediately.
- **Handshake Protocol**: The most complex. Allows client and server to:
    1. Establish security capabilities (protocol version, session ID, cipher suite, etc.).
    2. Server might send a certificate, request client's certificate, etc.
    3. Client sends its certificate (if requested), key exchange information, certificate verify if needed.
    4. Change cipher spec messages + finish the handshake.
- **Heartbeat Protocol**: Periodic signal to indicate a still-active connection. Vulnerable to the famous "Heartbleed" exploit if improperly implemented.

TLS Attacks

- *Handshake protocol* attacks.
- *Record/data protocol* attacks.
- *PKI* attacks (e.g., compromised CAs).
- *Heartbleed* (a bug in Heartbeat protocol implementation allowed reading random chunks of server memory).

HTTPS (HTTP over TLS)

- Standard HTTP requests/responses *over* a TLS-secured connection.
- When connection closes, TLS also closes the underlying TCP connection.

4.3 IP Security (IPsec)

- Provides security at the *network layer* for IPv4/IPv6.
- When implemented in a firewall/router, secures all cross-perimeter traffic.
- Transparent to applications and end users.

IPsec Sub-protocols

- **Authentication Header (AH)**: Data integrity and authentication of IP packets.
- **Encapsulating Security Payload (ESP)**: Encryption and optional authentication.
- **Internet Key Exchange (IKEv2)**: Key management and related functionalities.

Observation 5. *Modern IPsec usually omits AH (deprecated) because ESP can provide authentication as well.*

IPsec Modes

- **Transport mode**: End-to-end communication. ESP encrypts the payload but not the IP header.
- **Tunnel mode**: The entire IP packet is protected (commonly used for VPNs).

Definition 15 (Security Association)**.** *A one-way relationship between sender and receiver that provides security services to traffic flow. Defined by three parameters:*

1. *Security Parameter Index (SPI)*
2. *IP Destination Address*
3. *Protocol Identifier (AH or ESP)*

Two-way secure exchanges need two SAs (one for each direction).

4.3.1 Virtual Private Network (VPN)

Definition 16 (VPN)**.** *A* virtual network *built on top of an existing physical network infrastructure, able to provide secure communication (confidentiality, integrity, authentication) between endpoints.*

VPN Characteristics

- Security goals: confidentiality, integrity, authentication, replay protection, traffic analysis protection, etc.
- **Tunneling**: A virtual network connection on top of another network.
- *Host-to-Host*: Connects one host to another.
- *Host-to-Site*: Connects a single host to a network.
- *Site-to-Site*: Connects one network to another network.

Tunneling enables a **Protocol Data Unit (PDA)** to be transported from one site to another without its contents being processed by intermediate hosts on the route. The whole PDU gets encapsulated in another PDU sent on the network connecting the two sites. In particular, encapsulation takes place on the edge router of the source site, while decapsulation takes place on the edge router of the destination site.
Secure tunneling enables a PDU to be transported from one site to another without its contents being seen or changed by hosts on the route. This can be achieved by encrypting the PDU before its encapsulation.

SSL/TLS-based VPNs

- Offer
  - **Application** via two-factor authentication, X.509 certificates, etc.
  - **Encryption and integrity protection** via SSL/TLS VPN gateways.
  - **Access control**, which may be per-user, per-group or per-application.
  - **Endpoint security control** through validation of the security compliance of clients attempting to use the VPN.
  - **Intrusion prevention** evaluating decrypted data to prevent malicious attacks and malwares.
- Can be utilized to implement:
  - **Proxying** the intermediate device appears as true from the server to the client.
  - **Application translation** conversion of information from one protocol to another.
  - **Network extension** provision of partial or complete network access to remote users, typically via Tunnel VPN. It can be achieved through full tunneling, where all the network traffic goes through the tunnel, or through split tunneling, where the organization's traffic goes through the tunnel, while other traffic uses remote user's default gateway.

4.4 Anonymity with Tor

Definition 17 (Anonymity)**.** *The property of a subject to not be identifiable within a set of subjects, including* unlinkability *of action and identity, and* unobservability.

Limitations of Encryption/VPN for Anonymity

- Packets still have source/destination IP in headers.
- ISPs log routing info, accessible to law enforcement.
- VPN providers also keep logs, meaning they can reveal a user's identity.

Tor (The Onion Router) The idea behind Tor is based on the plain old *"ask someone else to send it for you"*:

- Based on Onion Routing: multiple layers of encryption, passing through several relays.
- Each relay peels off one layer and forwards the packet.
- Only the first relay knows the source; only the last relay knows the destination.
- Typically uses three relays:
  - **Guard (entry) node**: knows the source IP but not the destination.
  - **Middle relay**: knows neither source nor destination.
  - **Exit relay**: knows the destination but not the source.

Observation 6 (Observation 9)**.** *Tor does not provide encryption from the exit node to the final destination (unless the end-to-end connection is secured with TLS/HTTPS or similar).*

Examples

- **HTTP + Tor**: Packet headers are hidden from intermediate hosts except that the exit ISP sees the final destination. Everyone can see that Tor is in use.
- **HTTPS + Tor**: End-to-end encryption combined with Tor's routing anonymity. The final ISP sees the destination but *not* the content; still can see Tor is in use.

Observation 7 (Observation 10)**.** *Tor is limited to TCP streams and applications supporting SOCKS.*