Information Security

Course Outcomes:

1. Make use of security concepts and classical encryption techniques to establish secured communications. L3
2. Implement symmetric and asymmetric key algorithms for designing the security features for transfer of information from one end to another end. L3
3. Apply the features of authentication for acknowledging the two way communication process using message authentication codes. L3
4. Develop Privacy using SSL and TLS for constructing applications which use web services. L3
5. Develop security Mechanism across system and IP layers of communication. L3

Security Concepts

- OSI Architecture,
- Security attacks,
- Security services,
- Security mechanisms,
- A Model for Network Security.

- Information security is a broad field focused on protecting digital data, systems, and networks from unauthorized access, misuse, disclosure, destruction, or modification.
- Its main goals are to ensure confidentiality, integrity, and availability of information, often summarized by the CIA triad:

Security in OSI Layers:

- **Physical**: Physical access control, device hardening
- **Data Link**: MAC filtering, network access control
- **Network**: Firewalls, VPNs, IPSec
- **Transport**: TLS/SSL, secure ports
- **Session**: Session management, multi-factor authentication
- **Presentation**: Data encryption, encoding
- **Application**: Secure coding, patching, Web Application Firewalls

Security attacks

- Network Attacks
- Man-in-the-Middle
- It is a cyber security attack where an attacker secretly intercepts and manipulates communication between two parties without their knowledge.
- This allows the attacker to eavesdrop, alter messages, or impersonate one of the parties to steal sensitive information.
- Common MitM techniques include **Wi-Fi eavesdropping, ARP spoofing, and SSL stripping**.

- Denial of Service (DoS)

- A **Denial of Service (DoS) attack** is a type of cyber attack that aims to overwhelm a system, network, or website with excessive requests, making it unavailable to legitimate users.
- Attackers achieve this by **flooding the target with traffic or exploiting vulnerabilities** to crash services.

- IP Spoofing
- IP spoofing is the creation of Internet Protocol (IP) packets which have a modified source address in order to either hide the identity of the sender, to impersonate another computer system, or both.
-

- Session Hijacking
- **Session hijacking** is a cyberattack where an attacker takes control of an active user session by stealing session tokens or cookies.

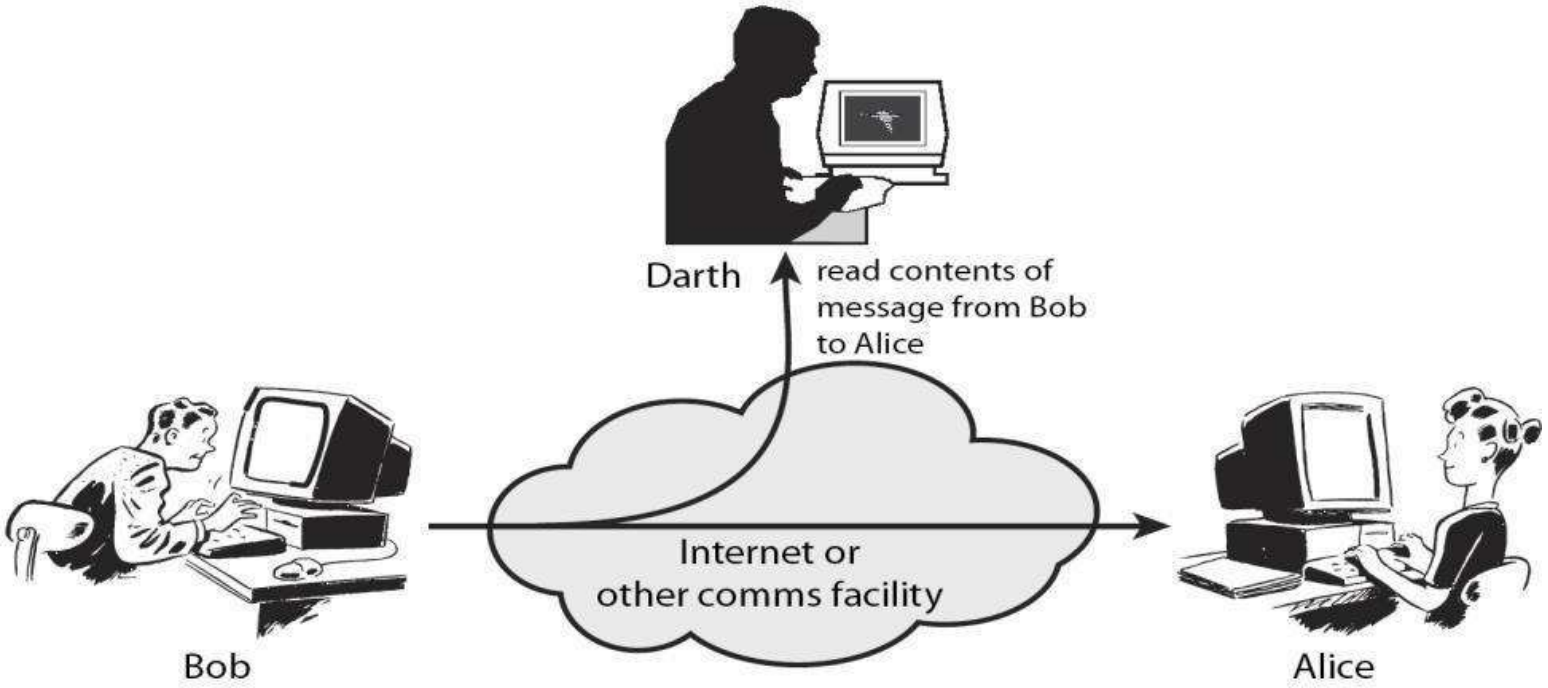
- **Application Attacks**
- SQL Injection
- SQL injection is a code injection technique that might destroy your database.
- SQL injection is the placement of malicious code in SQL statements, via web page input.
- Cross-Site Scripting (XSS)
- Cross-site scripting (XSS) is a type of web security vulnerability that allows attackers to inject malicious scripts into a website. This can lead to the theft of user data, session hijacking, and other consequences.
- Buffer Overflow
- buffer overflow occurs when the amount of data in the buffer exceeds its storage capacity. That extra data overflows into adjacent memory locations and corrupts or overwrites the data in those locations.

- Malware Attacks
- Viruses
- A computer virus is a type of malicious software, or malware, that spreads between computers and causes damage to data and software.
- Worms
- A computer worm is a type of malware that can automatically propagate or self-replicate without human interaction, enabling its spread to other computers across a network.
- Trojans
- Trojan" is a type of malware that typically gets hidden as an attachment in an email or a free-to-download file, then transfers onto the user's device.
- Ransom ware
- Ransomware is a type of malware attack in which the attacker locks and encrypts the victim's data, important files and then demands a payment to unlock and decrypt the data.

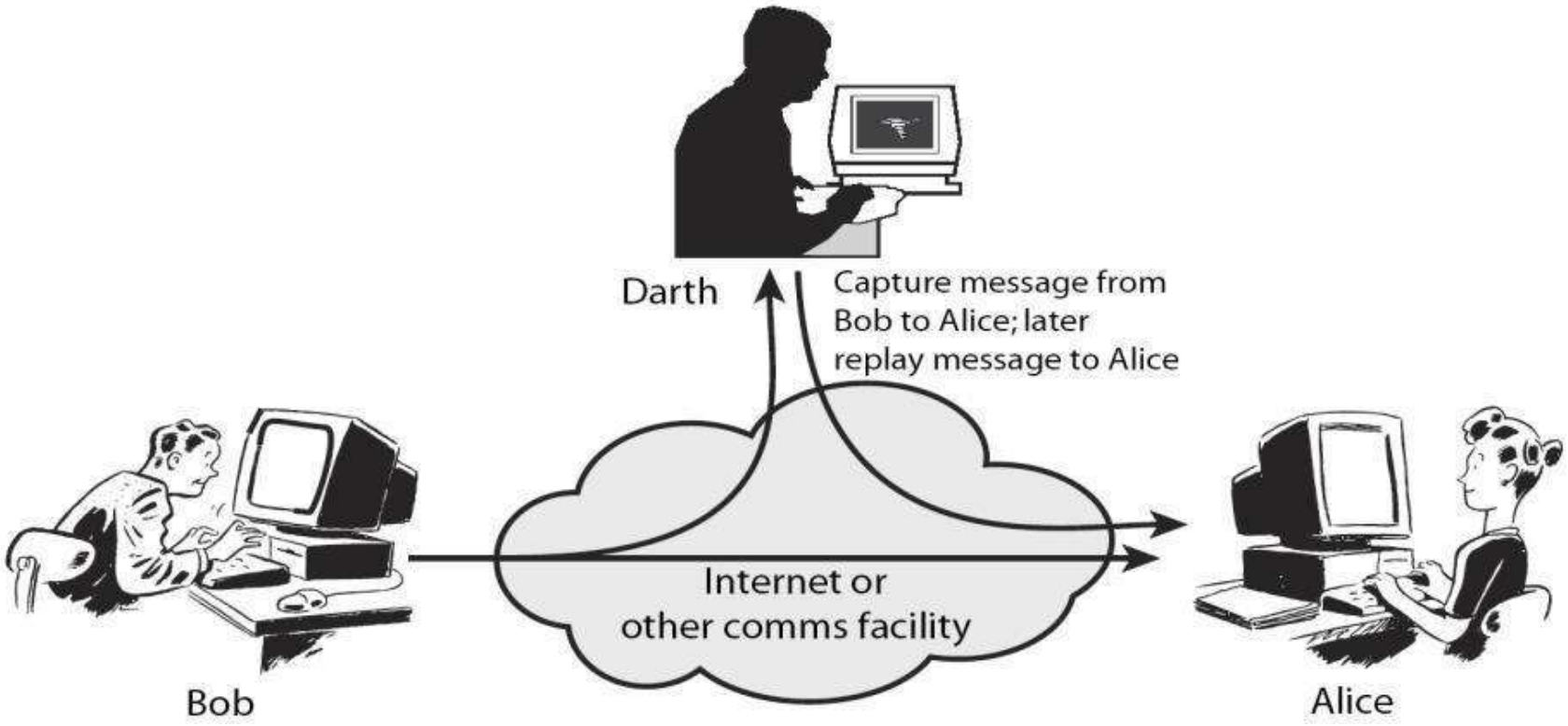
- Password Attacks
- Brute Force Attack
- A brute force attack is a cyberattack that involves trying many different combinations of usernames and passwords to gain access to a system.
- Credential Stuffing
- Credential based attacks occur when attackers steal credentials to gain access, bypass an organizations security measures, and steal critical data

Generic types of attacks

- Passive Attack



Active Attack



Security services & Mechanisms

- Security services in the context of information security are mechanisms and tools designed to protect data, systems, and networks from a wide range of threats.
- These services play a key role in ensuring the **confidentiality, integrity,** and **availability** (the CIA triad) of information across an organization.

Confidentiality Services

- **Encryption**: Protects data by converting it into an unreadable format (cipher text) that can only be converted back to its original form (plaintext) with a decryption key. This ensures that only authorized parties can access sensitive information.
  - **Example**: AES (Advanced Encryption Standard) for encrypting data at rest or in transit.
- **Access Control**: Ensures that only authorized users or systems can access specific resources or data.
  - **Example**: Role-based access control (RBAC), discretionary access control (DAC), and mandatory access control (MAC).

Integrity Services

- **Hashing**: Ensures data integrity by generating a unique fixed-size string (hash) that represents the original data. Even small changes in the data will result in a completely different hash value.
  - **Example**: MD5, SHA-256 (Secure Hash Algorithm).
- **Digital Signatures**: Provides proof of data integrity and authenticity by allowing the sender to sign a message with a private key, which can then be verified by the recipient using the sender's public key.
  - **Example**: RSA or ECDSA digital signatures.
- **Checksums**: Similar to hashing, checksums are used to detect errors in data transmission or storage, verifying that data has not been tampered with.

Availability Services

- **Redundancy**: Ensures that systems or data remain accessible by having backup components (e.g., duplicate hardware, alternative servers, or data replication) in place to replace failed parts.
  - **Example**: Using failover clusters or cloud services for automatic backup and recovery.
- **Load Balancing**: Distributes incoming traffic or requests across multiple servers or systems to prevent any single point of failure, ensuring consistent service availability.
  - **Example**: Using a load balancer to distribute traffic across web servers to ensure uptime during high traffic periods.

Model For Network Security

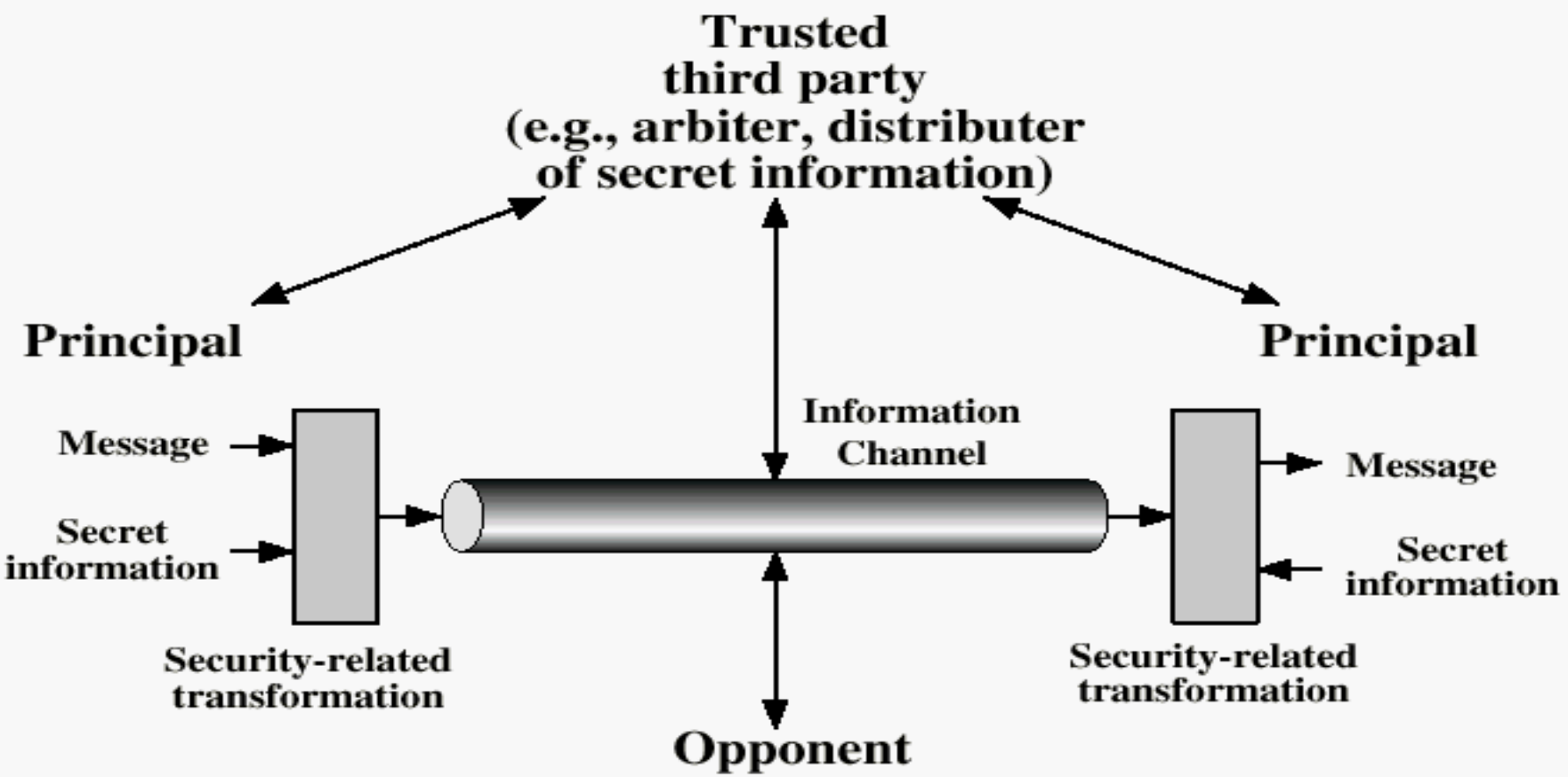


Figure 1.3 Model for Network Security

This general model shows that there are four basic tasks in designing a particular security service:

- Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
- Generate the secret information to be used with the algorithm.
- Develop methods for the distribution and sharing of the secret information.
- Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

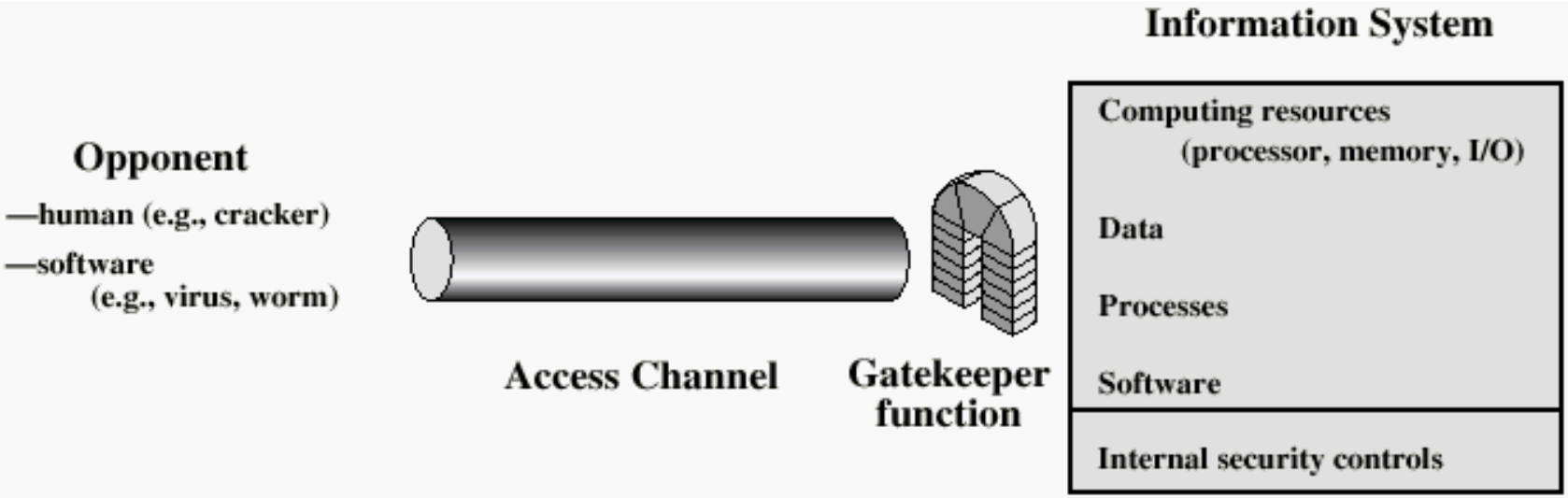


Figure 1.4 Network Access Security Model

Cryptography Concepts and Techniques

- **Cryptology** has two parts namely, **Cryptography** which focuses on creating secret codes and **Cryptanalysis** which is the study of the cryptographic algorithm and the breaking of those secret codes. The person practicing Cryptanalysis is called a **Cryptanalyst**.

Cryptographic systems are generally classified along 3 independent dimensions:

- Type of operations used for transforming plain text to cipher text
- Substitution, in which each element in the plaintext is mapped into another element, and Transposition, in which elements in the plaintext are rearranged.
- The number of keys used
- If the sender and receiver uses same key then it is s to be symmetric key (or) single key (or) conventional encryption. If the sender and receiver use different keys then it is said to be public key encryption
- The way in which the plain text is processed

A block cipher processes the input and block of elements at a time, producing output block for each input block. A Stream cipher processes the input elements continuously, producing output element one at a time, as it goes along.

SUBSTITUTION TECHNIQUES

- A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.
- If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.

CAESAR CIPHER

- The Caesar cipher method is based on a mono-alphabetic cipher and is also called a shift cipher or additive cipher.
- The Caesar cipher is a kind of replacement (substitution) cipher, where all letter of plain text is replaced by another letter.
- **Ciphertext:** It is an encrypted message after applying some technique.
- **The formula of encryption is:**
- $E_n (x) = (x + n) \bmod 26$
- **The formula of decryption is:**
- $D_n (x) = (xi - n) \bmod 26$
- If any case (Dn) value becomes negative (-ve), in this case, we will add 26 in the negative value.

Where,

- E denotes the encryption
  D denotes the decryption
  x denotes the letters value
  n denotes the key value (shift value)

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

- Use the Caesar cipher to encrypt and decrypt the message "HELLO," and the key (shift) value of this message is 15.
- **Encryption**
- We apply encryption formulas by character, based on alphabetical order.
- The formula of encryption is:
- **$E_n(x) = (x + n) \bmod 26$**

Plaintext: H → 07	E_n: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: E → 04	E_n: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: L → 11	E_n: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: L → 11	E_n: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: O → 14	E_n: (14 + 15) mod 26	Ciphertext: 03 → D

Note that the Caesar cipher is mono alphabetic, so the same plaintext letters are encrypted as the same letters. Like, "HELLO" has "L", encrypted by "A".

The encrypted message of this plain text is "WTAAD".

- **Decryption**
- We apply decryption formulas by character, based on alphabetical order.
- The formula of decryption is:
- **D_n (x) = (xi - n) mod 26**

Ciphertext: W → 22	D_n: (22 - 15) mod 26	Plaintext: 07 → H
Ciphertext: T → 19	D_n: (19 - 15) mod 26	Plaintext: 04 → E
Ciphertext: A → 00	D_n: (00 - 15) mod 26	Plaintext: 11 → L
Ciphertext: A → 00	D_n: (00 - 15) mod 26	Plaintext: 11 → L
Ciphertext: D → 03	D_n: (03 - 15) mod 26	Plaintext: 14 → O

The decrypted message is "HELLO".

Note: If any case (D_n) value becomes negative (-ve), in this case, we will add 26 in the negative value. Like, the third letter of the ciphertext.

D_n = (00 - 15) mod 26

= -15

The value of dn is negative, so 26 will be added to it.

= -15 + 26

= 11

Playfair cipher

- The **Playfair cipher** is a **manual symmetric encryption technique** ,It is a **digraph substitution cipher**, meaning it encrypts pairs of letters rather than single letters, making it more secure than simple substitution ciphers like the **Caesar cipher**.

How the Playfair Cipher Works

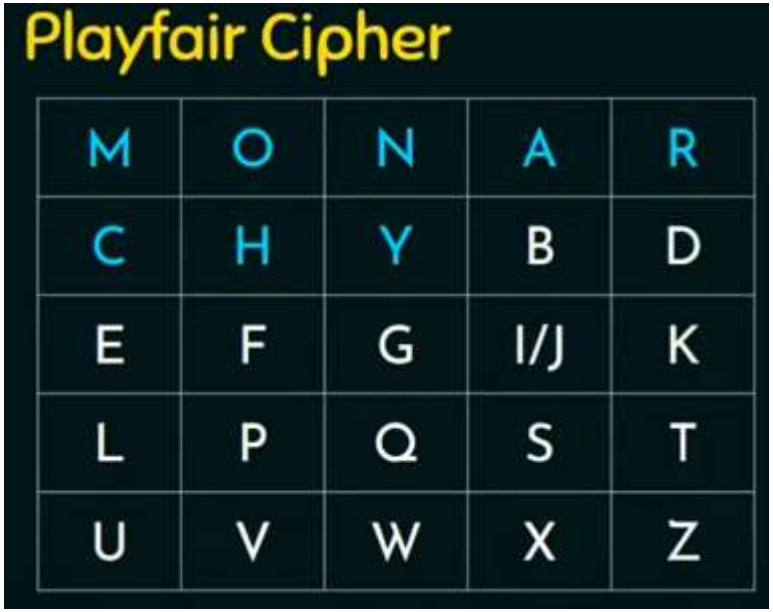
- **Key Square Generation**:
- A **5×5 grid** is created using a keyword (removing duplicate letters).
- The remaining letters of the alphabet are then filled in, typically merging **I** and **J** to fit 25 letters into the grid.
- **Plaintext Preparation**:
- The message is split into **letter pairs (digraphs)**.
- If a pair contains the same letter (like "EE"), an **X** is inserted between them.
- If the message has an odd number of letters, an **X** is added at the end.

- **Encryption Rules**:
- **Same Row:** Replace each letter with the letter **to its right** (looping to the start if needed).
- **Same Column:** Replace each letter with the letter **below** it (looping to the top if needed).
- **Rectangle Rule:** If the letters form a rectangle in the grid, swap them with the letters **on the same row but opposite corners**.
- **Decryption**:
- Reverse the encryption rules.

Example

- **Keyword: "MONARCHY"**

Plaintext: HELLO → HE LX LO
Using the Playfair rules, we get an encrypted text like **“CF US PM"**.



Hill cipher

- The **Hill cipher** is a **polygraphic substitution cipher** based on **linear algebra**. It was invented by **Lester S. Hill** in 1929 and uses **matrix multiplication** to encrypt blocks of letters.
- **Key Concepts**
- **Matrix-Based Encryption**
  - The cipher operates on **blocks** of letters (e.g., 2x2 or 3x3).
  - Letters are converted into **numbers** (A = 0, B = 1, ..., Z = 25).
  - A **key matrix** is chosen and used to encrypt plaintext.

- **Encryption Process**
  - Break plaintext into **n-letter blocks** (depending on the matrix size).
  - Convert letters into numbers.
  - Multiply the plaintext vector by the key matrix **mod 26**.
  - Convert the resulting numbers back to letters.
- **Decryption Process**
  - Compute the **inverse of the key matrix (mod 26)**.
  - Multiply the ciphertext vector by the **inverse key matrix mod 26**.
  - Convert numbers back to letters.

Example: 2×2 Hill Cipher

Step 1: Choose Key Matrix (Mod 26)

Let's use:

$$K = \begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$$

Ensure det(K) is invertible mod 26.

Step 2: Convert Plaintext ("HI") to Numbers

- H = 7, I = 8 → [7, 8]

Step 3: Multiply by Key Matrix

$$\begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} (3 \times 7 + 3 \times 8) \mod 26 \\ (2 \times 7 + 5 \times 8) \mod 26 \end{bmatrix} = \begin{bmatrix} 21 + 24 \mod 26 \\ 14 + 40 \mod 26 \end{bmatrix} = \begin{bmatrix} 19 \\ 2 \end{bmatrix}$$

Step 4: Convert Back to Letters
19 → T, 2 → C
Encrypted text: **"TC"**

Transposition Techniques

- A **transposition cipher** is a cryptographic technique where the positions of the letters in plaintext are **rearranged** according to a predefined pattern, without changing the actual letters.
- This differs from substitution ciphers, which replace letters with other symbols or letters.

Rail Fence Cipher

- **What is Rail Fence cipher? (Definition)**
- The **Rail Fence cipher** (also called **zigzag**) is a **transposition** cipher consisting in writing a text in **zig-zag** and reading it in a straight line.
- **How to encrypt using Rail Fence cipher?**
- **Rail Fence** Encryption uses an integer for the number of levels of the **zigzag**.
- The encoded message is written in **zig-zag** (like a **rail fence/**sawtooth) along a path with N levels/floors.

- Example: Encrypt DCODEZIGZAG with N=3 is writing

D - - - E - - - Z - -
- C – D – Z – G – A -
- - O - - - I - - - G

The cipher message is read by rows.
Example: The encrypted message is DEZCDZGAOIG

Columnar Transposition Cipher

- The Columnar Transposition Cipher is a form of transposition cipher just like Rail Fence Cipher. Columnar Transposition involves writing the plaintext out in rows, and then reading the ciphertext off in columns one by one.
- **Examples:**
- **Encryption** Input : Geeks for Geeks
- Key = HACK
- Output : e__kefGsGsrekoe_
- **Decryption** Input :
- e__kefGsGsrekoe_
- Key = HACK Output : Geeks for Geeks

- **Encryption**
- In a transposition cipher, the order of the alphabets is re-arranged to obtain the cipher-text.
- The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order.
- Width of the rows and the permutation of the columns are usually defined by a keyword.
- For example, the word HACK is of length 4 (so the rows are of length 4), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be “3 1 2 4”.
- Any spare spaces are filled with nulls or left blank or placed by a character (Example: _).
- Finally, the message is read off in columns, in the order specified by the keyword.

Encryption

Given text = Geeks for Geeks

Keyword = HACK **Length of Keyword** = 4 (no of rows) **Order of Alphabets in HACK** = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	_	f	o
r	_	G	e
e	k	s	_

Print Characters of column 1,2,3,4

Encrypted Text = e kefGsGsrekoe_

STEGANOGRAPHY

- A steganography technique involves hiding sensitive information within an ordinary, non-secret file or message, so that it will not be detected.
- The sensitive information will then be extracted from the ordinary file or message at its destination, thus avoiding detection.
- Steganography is an additional step that can be used in conjunction with encryption in order to conceal or protect data.

Different Types of Steganography

- Text Steganography
- Audio Steganography
- Video Steganography
- Image Steganography
- Network Steganography

Applications of Steganography

- **Covert Communication**: Securely transmitting secret messages.
- **Digital Watermarking**: Protecting copyright by embedding ownership data in media files.
- **Cybersecurity**: Hiding commands in malware or data exfiltration techniques.

Steganography vs. Cryptography

Feature	Steganography	Cryptography
Goal	Hides existence of message	Encrypts message to make it unreadable
Detectability	Hard to detect	Easily noticeable when encrypted
Security	Relies on secrecy of method	Relies on strength of encryption algorithm
Combination	Can be used with encryption for added security	Can complement steganography

Unit -II

Symmetric Key Ciphers:

Block Cipher principles, DES, AES, Block Cipher Modes of Operation, Stream ciphers, RC4.

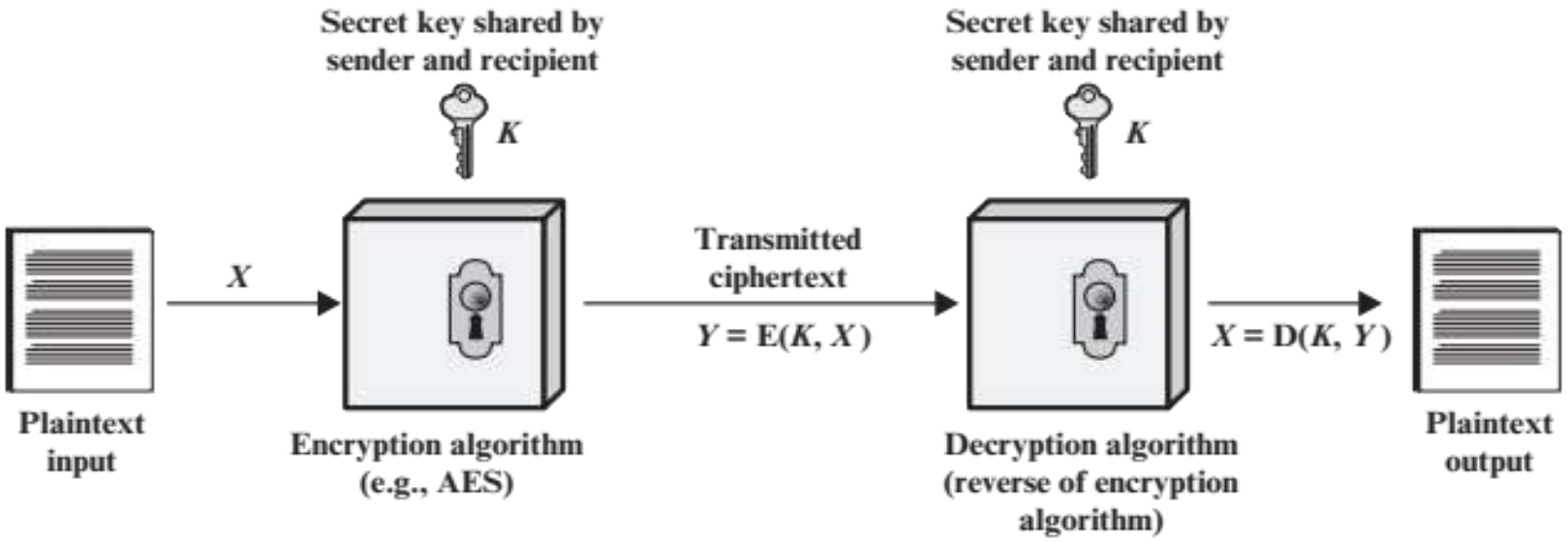
Asymmetric Key Ciphers:

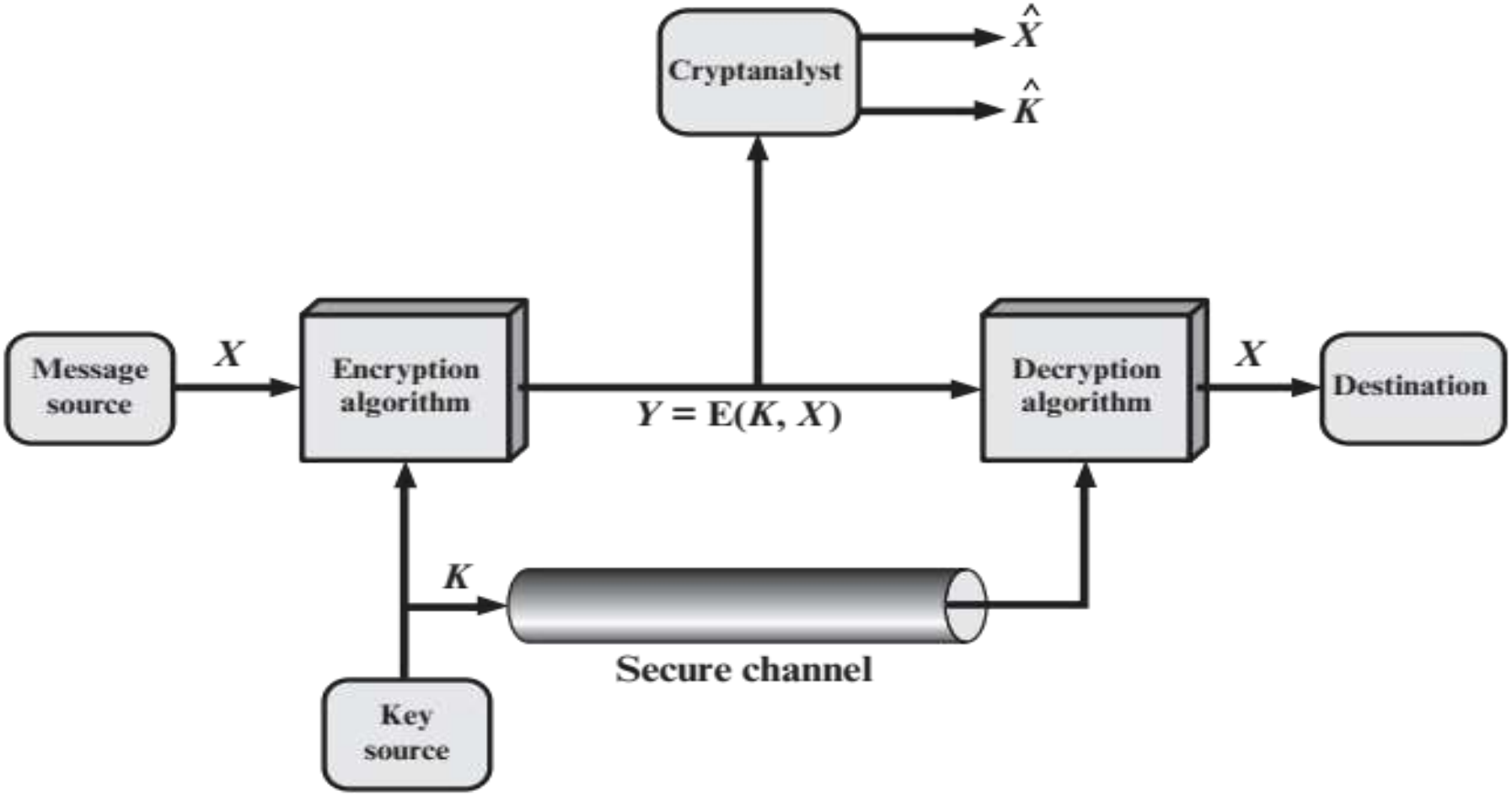
Principles of public key cryptosystems, RSA algorithm, Difffie Hellman Key Exchange, Elliptic Curve Cryptography and Arithmetic.

Symmetric Key Ciphers:

- A symmetric encryption scheme has five ingredients
- Plain Text
- Encryption Algorithm
- Secret Key
- Cypher Text
- Decryption Algorithm

Model of Symmetric Encryption





Model of Symmetric Cryptosystem

Block Cipher principles

- A block cipher is a method of encrypting data in blocks to produce cipher_text using a cryptographic key and algorithm.
- The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time.
- Most modern block ciphers are designed to encrypt data in fixed-size blocks of either 64 or 128 bits.

- Block ciphers are invertible symmetric key cryptographic algorithms.
- Here, *symmetric* means that the same cryptographic key is used to both encrypt and decrypt messages.
- The key is used to transform (encrypt) one block of information at a time.

- **Block cipher works**:
- It breaks down the plaintext input into blocks of fixed length, e.g., 64 bits or 128 bits.
- If the plaintext length is not a multiple of the block size, it is padded with redundant information to provide a complete final block.
- Each block is encrypted using the cipher key.
- The ciphertext is generated by connecting the encrypted blocks.
- The message recipient receives the same key and uses it to decrypt the message into the original plaintext.

Feistel Block Cipher

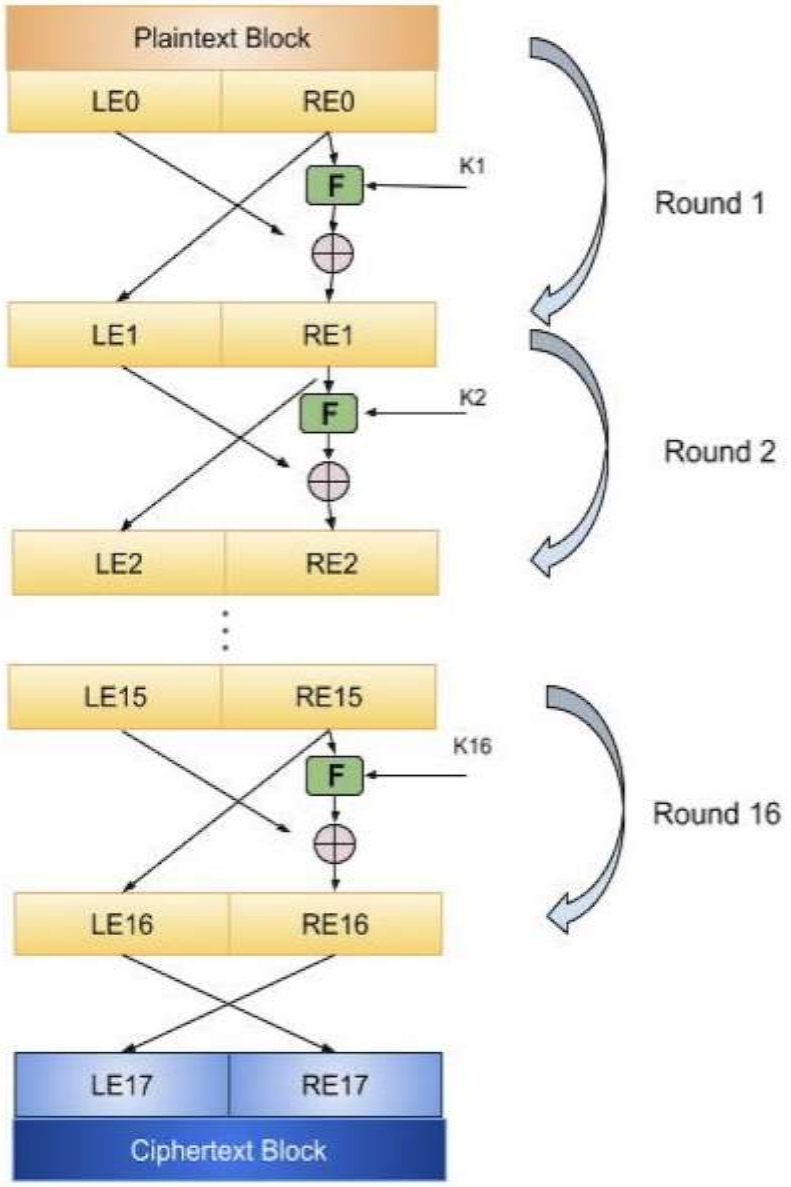
- A framework or design model called the Feistel cipher which is used to create different symmetric block ciphers, including DES.
- The Feistel structure demonstrates the implementation processes of confusion and diffusion.
- Using a substitution method, confusion creates a complex relationship between the encryption key and the ciphertext. But diffusion uses a permutation process for creating a complex link between plaintext and cipher text.

- Algorithm
- Make a list of every character in plaintext.
- After converting the plaintext to ascii, format it in 8-bit binary.
- Separate the binary plaintext string into its left (L1) and right (R1) parts.
- For each of the two rounds, generate two random binary keys (K1 and K2), each of equal length to the plaintext.

Encryption

Step 1 – The plaintext is split up into fixed-size blocks, and only one block is handled at a time in this initial phase. Two inputs for the encryption technique are a block of plaintext and a key K.

Step 2 – Split the block of plaintext into two parts. The plaintext block will have two distinct representations: LE0 for the left half of the block and RE0 for the right half. To create the ciphertext block, the two parts of the plaintext block (LE0 and RE0) will undergo multiple rounds of plaintext processing.

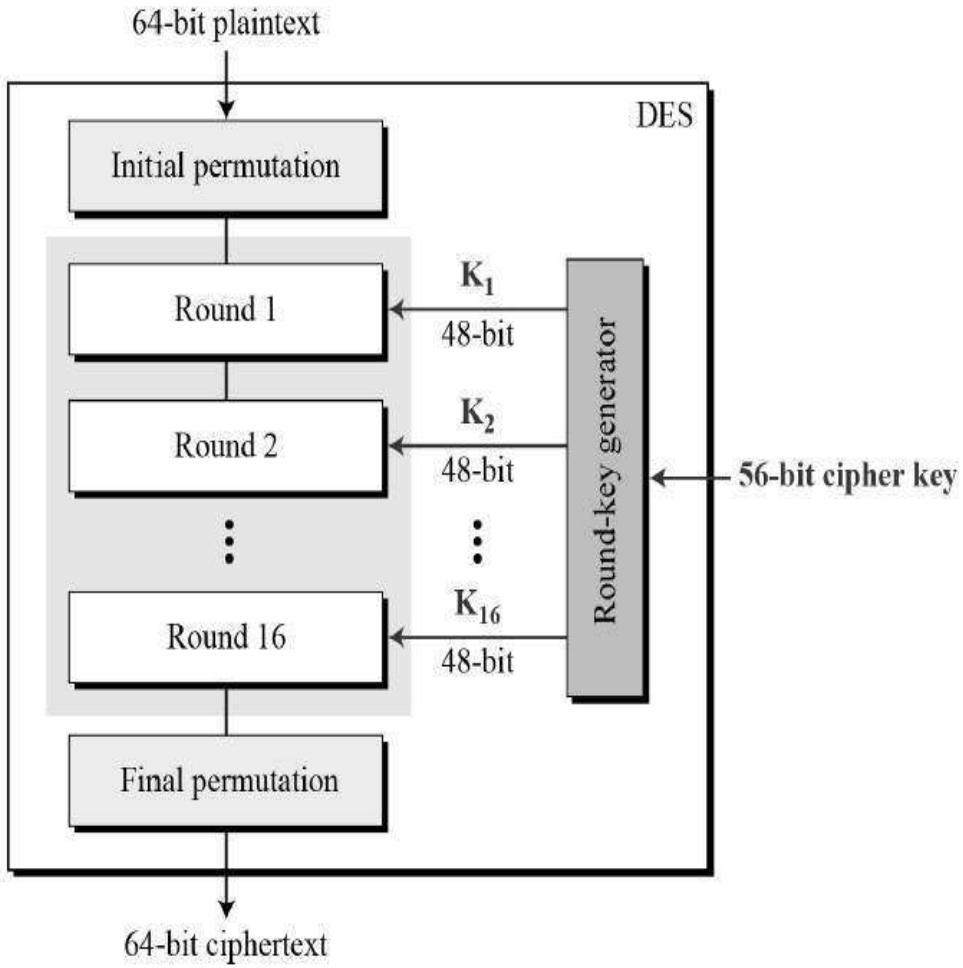


Data Encryption Standard algorithm

- The DES which stands for Data Encryption Standard algorithm, is a symmetric key block cipher created by an IBM team in the early 1970s and some time later it is adopted by the National Institute of Standards and Technology.
- In this method we divides plaintext into 64-bit blocks and transforms it to ciphertext with the help of 48-bit keys.

Structure of DES

- DES uses 16 round Feistel structure.
- The block size is 64-bit.
- Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

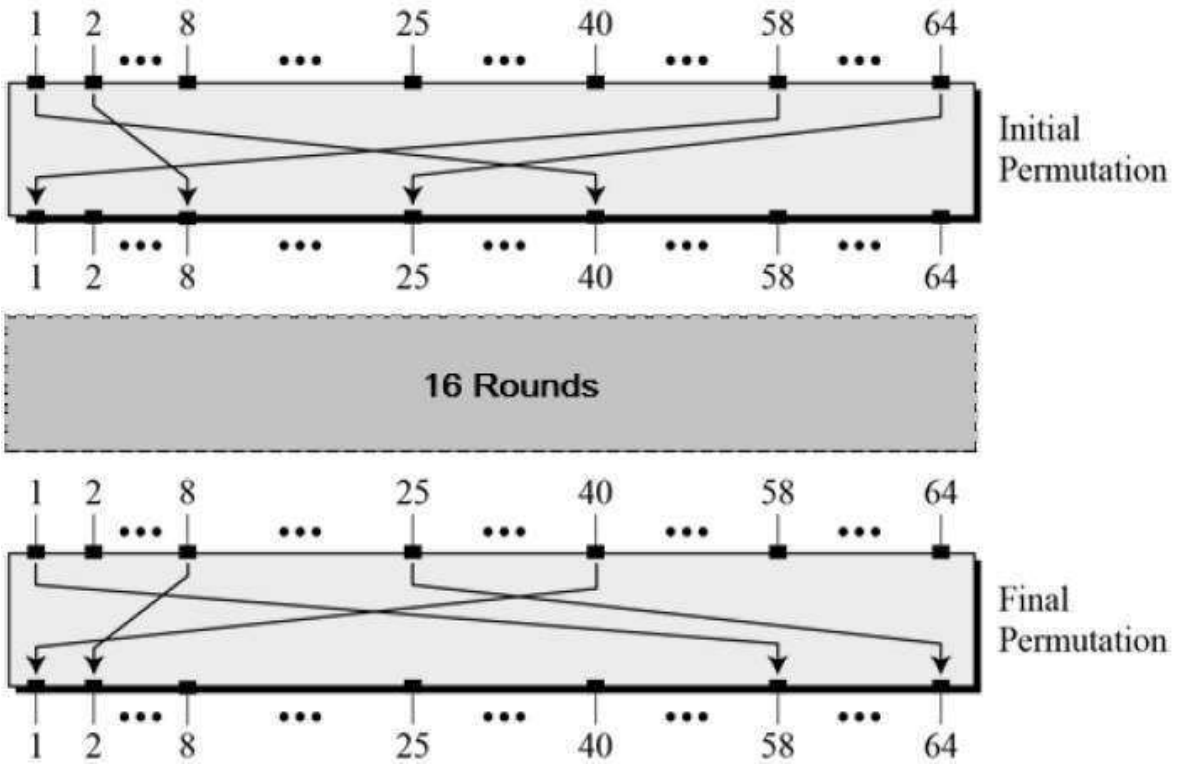


As DES is based on the Feistel Cipher, all that is required to specify DES is –

- Round function
- Key schedule
- Any additional processing - Initial and final permutation

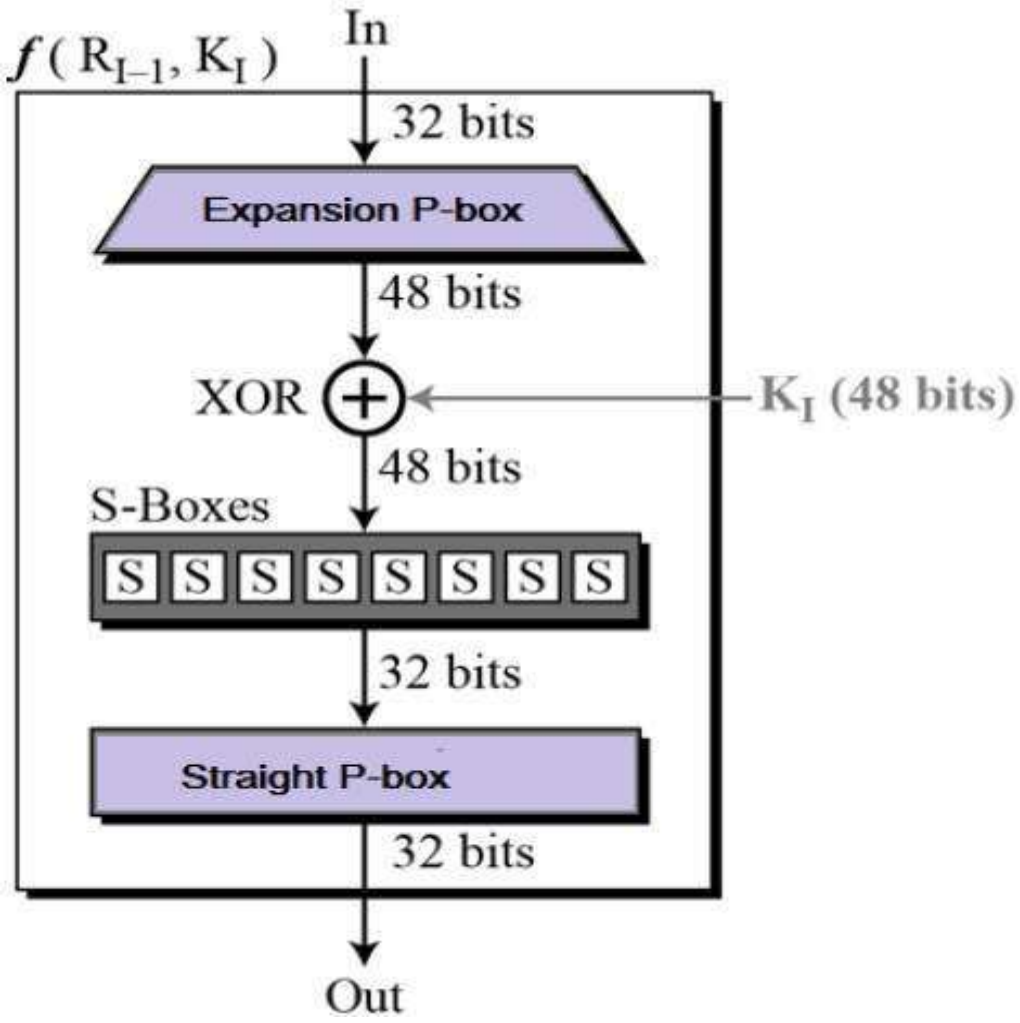
Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows −



Round Function

The heart of this cipher is the DES function, f. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



- Expansion Permutation Box – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted.
- The graphically depicted permutation logic is generally described as table in DES specification illustrated
- **XOR (Whitener)** – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. This is the only type of operation that uses the round key.
- **Substitution Boxes** – The S-boxes carry out the real mixing (confusion). Eight S-boxes, each with a 6-bit input and a 4-bit output, are used in DES.
- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32 bit section.
- **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation with rule

Algorithm of DES

- **Step 1** – The 64-bit plaintext block is initially provided to an initial permutation (IP) function to start the process.
- **Step 2** – After that, the plaintext undergoes to the initial permutation (IP).
- **Step 3** – The first permutation (IP) then generates Left Plain Text (LPT) and Right Plain Text (RPT), which are the two sides of the permuted block.
- **Step 4** – The encryption procedure consists of sixteen cycles for each LPT and RPT.
- **Step 5** – Lastly, the LPT and RPT become one, and the newly combined block is subjected to a Final Permutation (FP).
- **Step 6** – This procedure yields the intended 64-bit ciphertext.

NOTE:- Step 4 of the encryption process can be further divided into five stages – Key transformation, Expansion permutation, S-Box permutation, P-Box permutation, XOR and swap.

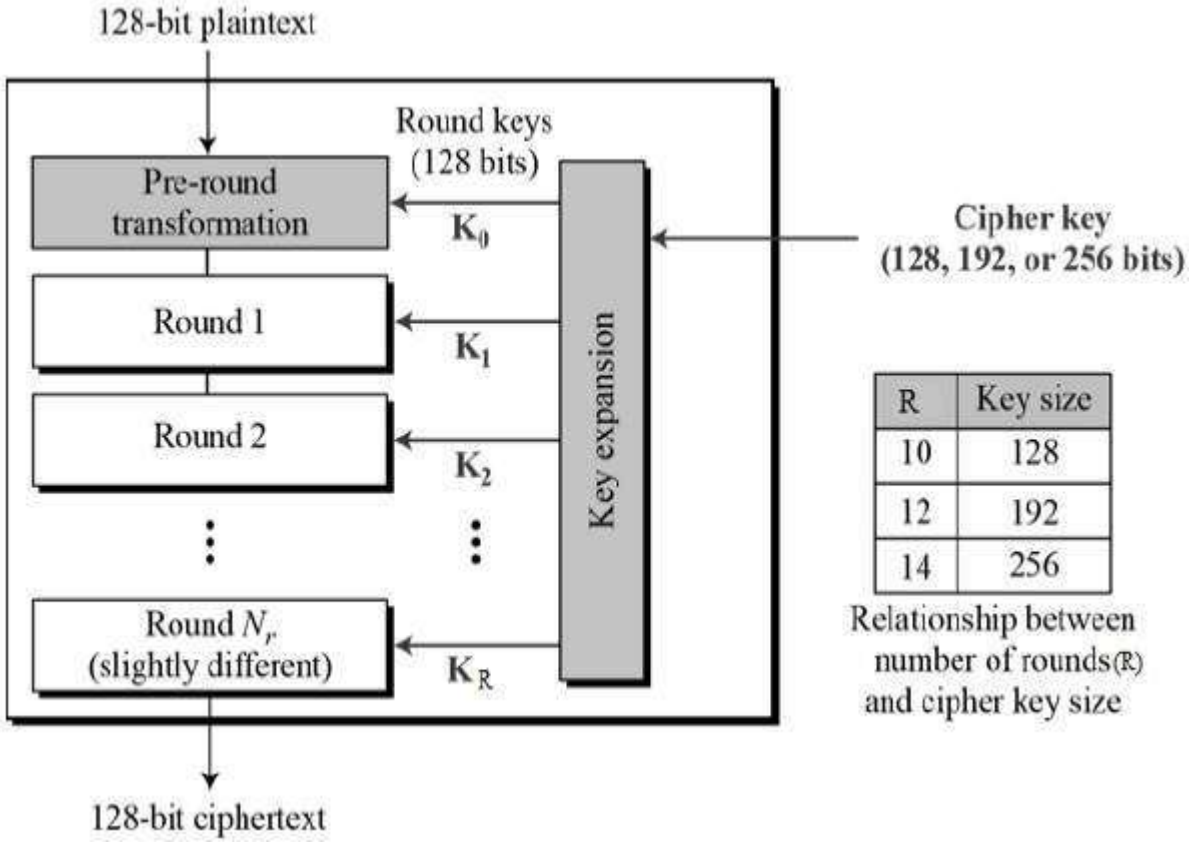
Advanced Encryption Standard (AES)

- Advanced Encryption Standard (AES) is a highly trusted **encryption algorithm** used to secure data by converting it into an unreadable format without the proper key.
- It is widely used today as it is much stronger than DES and triple DES despite being harder to implement.
- **AES encryption** uses various **key lengths** (128, 192, or 256 bits) to provide strong protection against unauthorized access.
- This **data security** measure is efficient and widely implemented in securing **internet communication**, protecting **sensitive data**, and encrypting files.
- AES, a cornerstone of modern cryptography, is recognized globally for its ability to keep information safe from cyber threats.

- AES is a Block Cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.
- AES relies on the substitution-permutation network principle, which is performed using a series of linked operations that involve replacing and shuffling the input data.

The schematic of AES structure

- AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys.
- Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

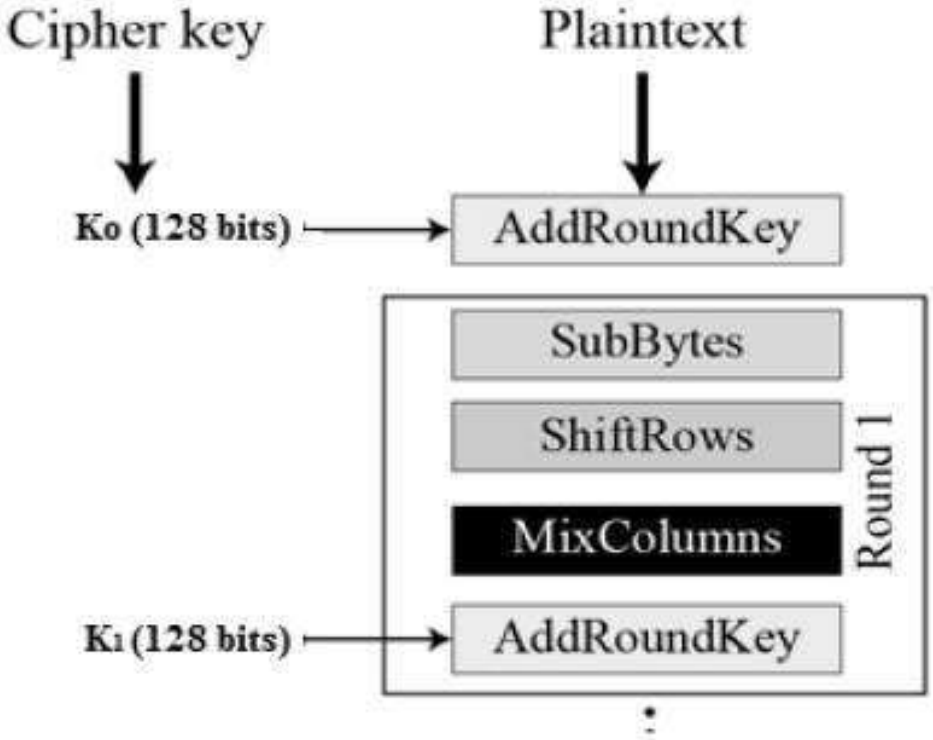


R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Encryption Process

- Here, we restrict to description of a typical round of AES encryption. Each round comprise of four sub-processes. The first round process is depicted below –



- **Byte Substitution (SubBytes)**
- The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.
- **Shift rows**
- Each of the four rows of the matrix is shifted to the left. Any entries that ‘fall off’ are re-inserted on the right side of row.

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

- MixColumns
- Each column of four bytes is now transformed using a special mathematical function.
- This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column.
- The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

- Addroundkey
- The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key.
- If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

Decryption Process

- The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order –
- Add round key
- Mix columns
- Shift rows
- Byte substitution
- Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

Block Cipher Modes of Operation

- Block Cipher Modes of Operation define how to securely encrypt and decrypt large amounts of data using a block cipher.
- A block cipher is an encryption algorithm that processes data in fixed-size blocks (e.g., 128 bits) rather than one bit at a time.
- However, to encrypt data larger than a single block, different modes of operation are used to ensure both security and efficiency. Here are a few common modes.

- Block cipher processes are classified into five categories
- **ECB Mode** (Electronic Code Block) – ECB is the simplest block cipher to operate. As each block of input plaintext is directly encrypted and the output is in the form of blocks of encrypted ciphertext.
- **CBC Mode** (Cipher Block Chaining) – Since ECB compromises certain security requirements, such as a direct connection between the cipher text and the plain text that makes it easier for attackers to decrypt the encoded information, CBC is the advanced mode of ECB.
- **CFB Mode** (Cipher Feedback) – In this case, the cipher is sent as feedback for the subsequent encryption block.
- **OFB Mode** (Output Feedback) – With the exception of sending the encrypted output as feedback rather than the actual cipher, which is the XOR output, the OFB operates in a manner very similar to that of the CFB.
- **CTR Mode** (Counter) – An implementation of a basic counter-based block cipher is the CTR.

- **Applications of Block Ciphers**
- Data Encryption:
- File and Disk Encryption:
- Virtual Private Networks (VPN):
- Secure Sockets Layer (SSL) and Transport Layer Security (TLS):
- Digital Signatures:

Initialization Vector (IV)

- **Initialization Vector (IV)** is a crucial concept in cryptography, especially when using block cipher modes of operation.
- It is a random or unique value used in combination with a secret key to initialize the encryption process.
- It ensures that the same plaintext encrypted multiple times will produce different ciphertexts, enhancing security by preventing patterns from emerging.

- **Importance of IV**
- **Initialization Vector (IV)** is a non-secret, unique value used to ensure that encrypting the same plaintext with the same key results in different ciphertexts.
- IVs are essential for maintaining the security and integrity of encryption processes, especially in block cipher modes like CBC, CFB, OFB, and CTR.
- Proper generation and management of IVs ensures they are unique and unpredictable and are vital to preventing cryptographic vulnerabilities.
- IVs should be transmitted or stored alongside the ciphertext to enable successful decryption, but they must never be reused with the same key.

Stream ciphers

- A stream cipher is an encryption method in which data is encrypted one byte at a time.
- In this method, a cryptographic key and algorithm are applied to each binary digit in a plaintext data stream to produce cipher_text.
- Stream ciphers make use of a common key (“symmetric key”) to code their data. Encryption and decryption processes of the data are handled by this symmetric key.

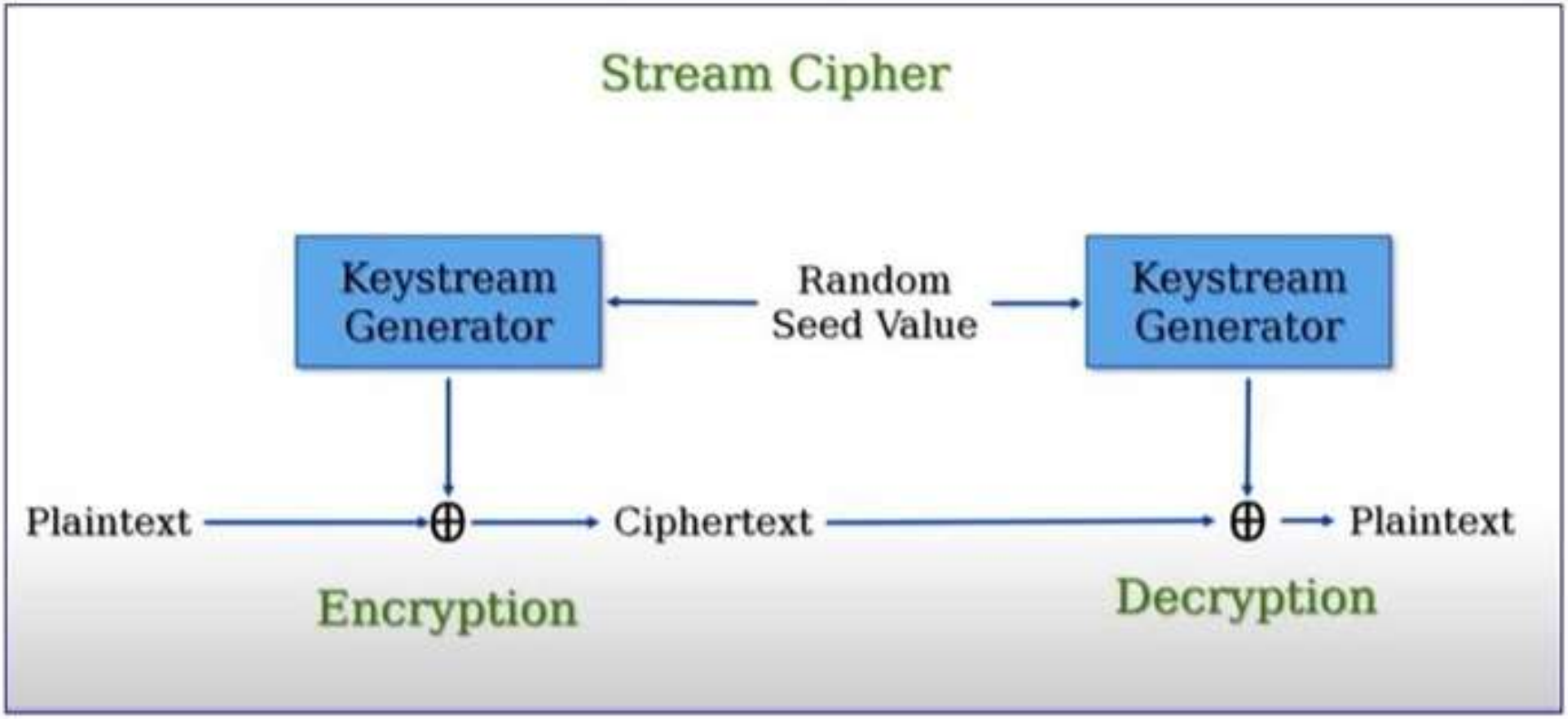
- **Plaintext** − The original message to be encrypted.
- **Keystreams** − Random sequences of characters (e.g., numbers, letters, symbols) that replace the plaintext characters.
- **Ciphertext** − The encrypted message. Key generation is a complex mathematical operation, but modern computers can perform it quickly.

- In a stream cipher, individual bits of plaintext are inputted and subjected to a mathematical operation. The result is jumbled text which needs the right key to decode.

- Using the proper key, the receiver can reverse the process and convert the scrambled text back to its plaintext.

- In stream cipher encryption, the key known as a one-time pad is exceptionally secure due to its unique property.

- It is designed to be equivalent in length to the message being encrypted, ensuring that an attacker cannot mathematically decipher the message without having the original key.

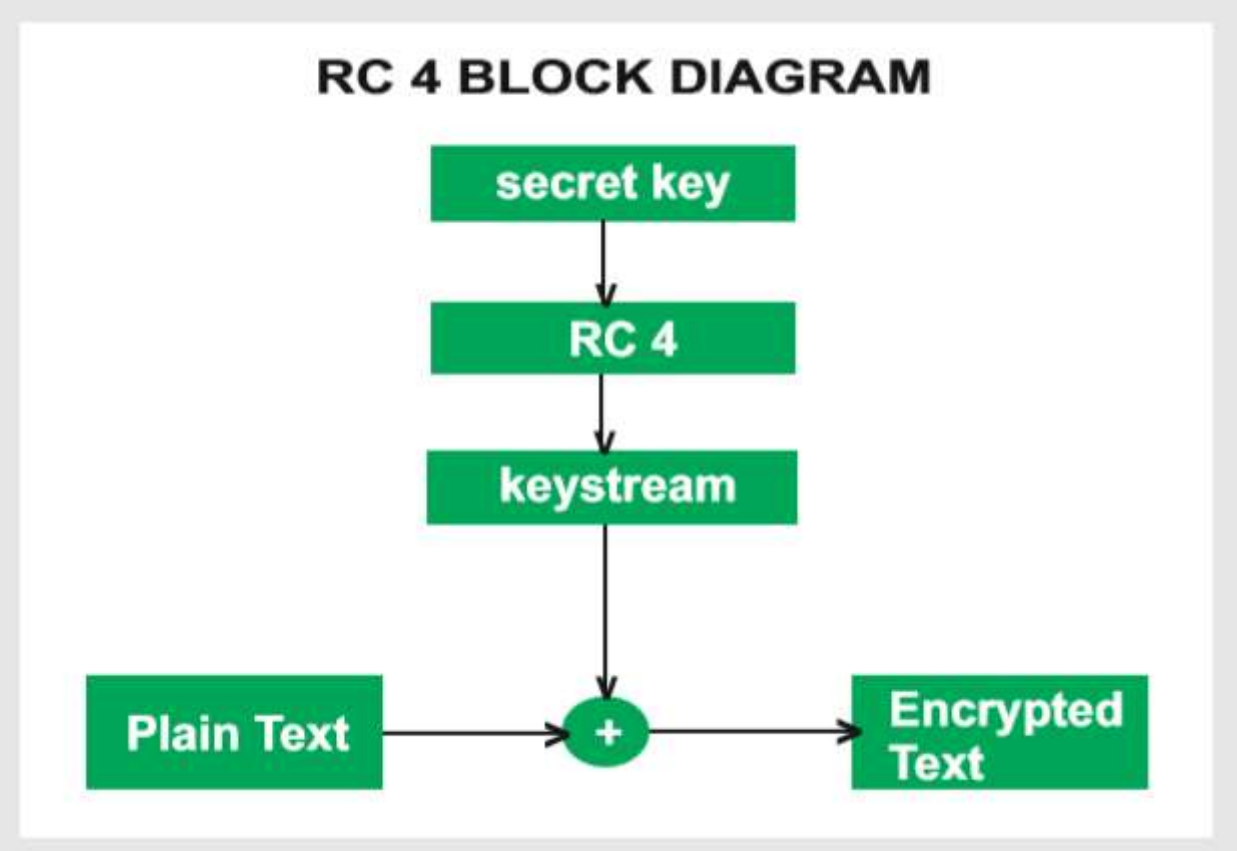

Stream Cipher
Keystream Generator
Random Seed Value
Keystream Generator
Plaintext
Ciphertext
Plaintext
Encryption
Decryption

- Keystream Generation
- Let us see at a basic example of keystream creation with the help of an XOR-based stream cipher.
- Let's say we have the below data −
- Key: 101011
- Initialization Vector (IV): 110100
- To create a stream of encrypted data (keystream):
- Set up the encryption algorithm with a secret key and initialization vector (IV).
- If needed, adjust the key and IV to be the same length as the message being encrypted.
- Combine the key and IV using an exclusive OR (XOR) operation to generate the keystream.

- Here is the step by step process −
- Key: 101011
- IV: 110100
- Keystream: 011111
- Now, let us say we have a plaintext message as: 1100101.
- To encrypt this plaintext using the keystream −
- Plaintext: 1100101
- Keystream: 011111
- Ciphertext: 1111010
- To decrypt the ciphertext, we would use the same keystream −
- Ciphertext: 1111010
- Keystream: 011111
- Plaintext: 1100101
- This example explains how to generate a keystream, which is then used to encode and decode a message, using a key and an initialization vector. For better safety, more complex methods and bigger key sizes are used in real-world applications.

RC4 Encryption

- RC4 means Rivest Cipher 4 invented by Ron Rivest in 1987 for RSA Security.
- It is a Stream Ciphers.
- Stream Ciphers operate on a stream of data byte by byte.
- RC4 stream cipher is one of the most widely used stream ciphers because of its simplicity and speed of operation.
- It is a variable key-size stream cipher with byte-oriented operations.
- It uses either 64 bit or 128-bit key sizes.
- It is generally used in applications such as Secure Socket Layer (SSL), Transport Layer Security (TLS), and also used in IEEE 802.11 wireless LAN std.


RC 4 BLOCK DIAGRAM
secret key
RC 4
keystream
Plain Text
+
Encrypted Text

- **KSA algorithm-**
- KSA is going to use the secret key to scramble this array. KSA is a simple loop, in which we are having two variable i and j. We are using these variables to rearrange the array. Rearranging the array is done by using a secret key.
- PRGA(Pseudo Random Generation Algorithm).

Encryption Procedure

- The user inputs a plain text file and a secret key.
- The encryption engine then generates the keystream by using KSA and PRGA Algorithm..
- This keystream is now XOR with the plain text, this XORing is done byte by byte to produce the encrypted text.
- The encrypted text is then sent to the intended receiver, the intended receiver will then decrypted the text and after decryption, the receiver will get the original plain text.

Decryption Procedure

- Decryption is achieved by doing the same byte-wise X-OR operation on the Ciphertext.

Asymmetric Key Ciphers

- Asymmetric encryption, also known as public-key cryptography, is a type of encryption that uses a pair of keys to encrypt and decrypt data.
- The pair of keys includes a public key, which can be shared with anyone, and a private key, which is kept secret by the owner.

- **Public key cryptography** has become an essential means of providing confidentiality, especially through its need of key distribution, where users seeking private connection exchange encryption keys.
- It also features digital signatures which enable users to sign keys to check their identities.
- **Public key Cryptosystem** – Asymmetric algorithms depends on one key for encryption and a distinct but related key for decryption.

A public key encryption scheme

- **Plaintext** − This is the readable message or information that is informer into the algorithm as input.
- **Encryption algorithm** − The encryption algorithm performs several conversion on the plaintext.
- **Public and Private keys** − This is a set of keys that have been selected so that if one can be used for encryption, and the other can be used for decryption.
- **Ciphertext** − This is scrambled message generated as output. It based on the plaintext and the key. For a given message, there are two specific keys will create two different ciphertexts.
- **Decryption Algorithm** − This algorithm get the ciphertext and the matching key and create the original plaintext.

RSA algorithm

- The RSA algorithm (**Rivest-Shamir-Adleman**) is a public key cryptosystem that uses a pair of keys to secure digital communication and transactions over insecure networks, such as the internet.
- RSA is a type of asymmetric encryption that uses two different but linked keys. In RSA cryptography, both the public and the private keys can encrypt a message. The opposite key from the one used to encrypt a message is used to decrypt it.

How does the RSA algorithm work?

- Use of the RSA algorithm typically consists of four stages: key generation, key distribution, encryption and decryption:
- **Key generation.** Two large prime numbers are selected and used to generate the public and private keys.
- **Key distribution.** The public key can be shared with anyone who needs to send encrypted messages to the recipient. The private key is kept secure and only known to the recipient.
- **Encryption.** To encrypt a message, the sender uses the recipient's public key to transform the message into ciphertext. That makes the message unreadable to anyone who doesn't have the private key.
- **Decryption.** The recipient uses their private key to decrypt the ciphertext back into the original message. That way, only the intended recipient is able to read the message.

Algorithm

The RSA algorithm holds the following features –

- RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
- The integers used by this method are sufficiently large making it difficult to solve.
- There are two sets of keys in this algorithm: private key and public key.

- Step 1: Generate the RSA modulus
- The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown –
- N=p*q Here, let N be the specified large number.
- Step 2: Derived Number (e)
- Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1).
- The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1

- Step 3: Public key
- The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.
- Step 4: Private Key
- Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows –
- $ed = 1 \bmod (p-1)(q-1)$

- Encryption Formula
- Consider a sender who sends the plain text message to someone whose public key is **(n,e).**
- To encrypt the plain text message in the given scenario, use the following syntax −

$$C = Pe \bmod n$$

- Decryption Formula
- The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver **C** has the private key **d**, the result modulus will be calculated as −

$$Plaintext = Cd \bmod n$$

Diffie-Hellman algorithm

- The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.

- we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.

- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Step-by-Step explanation is as follows:

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x = G^a mod P$	Key generated = $y = G^b mod P$
Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a mod P$	Generated Secret Key = $k_b = x^b mod P$
Algebraically, it can be shown that $k_a = k_b$	
Users now have a symmetric secret key to encrypt	

Example:

Step 1: Alice and Bob get public numbers P = 23, G = 9

Step 2: Alice selected a private key a = 4 and
Bob selected a private key b = 3

Step 3: Alice and Bob compute public values
Alice: x =(9^4 mod 23) = (6561 mod 23) = 6
Bob: y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Alice and Bob exchange public numbers

Step 5: Alice receives public key y =16 and
Bob receives public key x = 6

Step 6: Alice and Bob compute symmetric keys
Alice: ka = y^a mod p = 65536 mod 23 = 9
Bob: kb = x^b mod p = 216 mod 23 = 9

Step 7: 9 is the shared secret.

Elliptic Curve Cryptography and Arithmetic

- Elliptical curve cryptography (ECC) is a public key encryption technique based on elliptic curve theory that can be used to create faster, smaller and more efficient cryptographic keys.

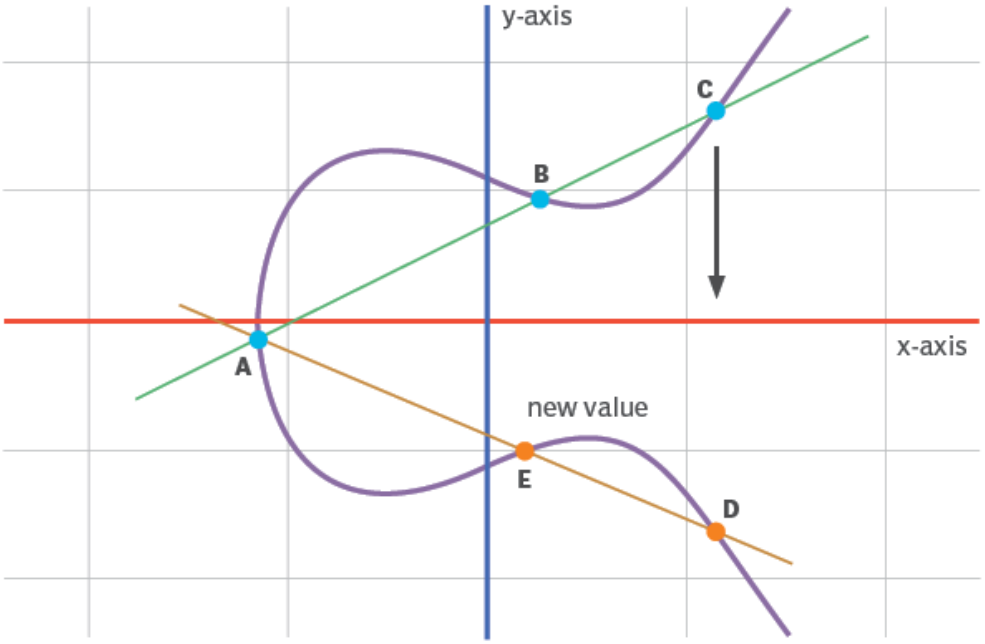
- ECC is an alternative to the Rivest-Shamir-Adleman (RSA) cryptographic algorithm and is most often used for digital signatures in cryptocurrencies, such as Bitcoin and Ethereum, as well as one-way encryption of emails, data and software.

- An elliptic curve is not an ellipse, or oval shape, but it is represented as a looping line intersecting two axes, which are lines on a graph used to indicate the position of a point. The curve is completely symmetric, or mirrored, along the x-axis of the graph.

- Public key cryptography systems, like ECC, use a mathematical process to merge two distinct keys and then use the output to encrypt and decrypt data. One is a public key that is known to anyone, and the other is a private_key that is only known by the sender and receiver of the data.
- ECC generates keys through the properties of an elliptic curve equation instead of the traditional method of generation as the product of large prime numbers. From a cryptographic perspective, the points along the graph can be formulated using the following equation:

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

Elliptical curve showing three points of intersection



SOURCE: ANDREW FROEHLICH

©2021 TECHTARGET. ALL RIGHTS RESERVED TechTarget

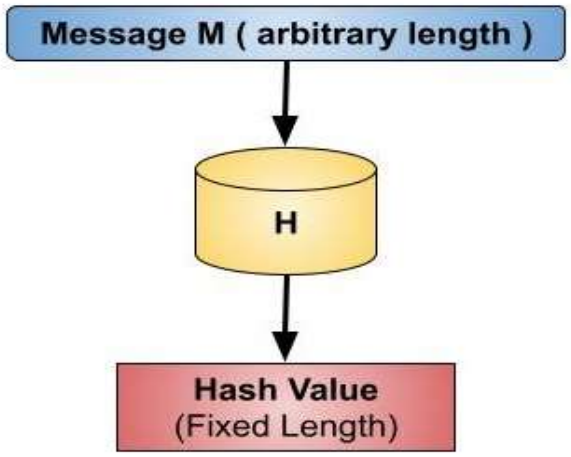
CRYPTOGRAPHIC HASH FUNCTIONS

Authentication requirements and Functions, Message Authentication Code, Secure Hash Algorithm (SHA- 512), Message authentication codes: HMAC, CMAC, Digital signatures,

AUTHENTICATION APPLICATIONS

Kerberos, X.509 Authentication Service Public Key Infrastructure.

- A hash function in cryptography is like a mathematical function that takes various inputs, like messages or data, and transforms them into fixed-length strings of characters. Means the input to the hash function is of any length but output is always of fixed length.
- Hash functions play a crucial role in various security applications, including password storage (hash values instead of passwords), digital signatures, and data integrity checks. Hash values, or message digests, are values that a hash function returns.



Authentication requirements and Functions

- Disclosure
- Traffic analysis
- Masquerade
- Content modification
- Sequence modification
- Timing modification
- Source repudiation
- Destination repudiation

- **Disclosure** : Release of message contents to any person or process not possessing the appropriate cryptographic key.
- **Traffic analysis** :
- Discovery of the pattern of traffic between parties.
- In a connection oriented application, the frequency and duration of connections could be determined.
- In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

Masquerade:

- Insertion of messages into the network from a fraudulent source.
- includes the creation of messages by an opponent that are purported to come from an authorized entity.
- Also included are fraudulent acknowledgments of message receipt or non receipt by someone other than the message recipient.

Content modification

- Changes to the contents of a message, including insertion, deletion, transposition, and modification.

Sequence modification

- Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

Timing modification

- Delay or replay of messages.
- In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.
- In a connectionless application, an individual message (e.g:- datagram) could be delayed or replayed.

Source repudiation –

- Denial of transmission of message by source.

Destination repudiation:

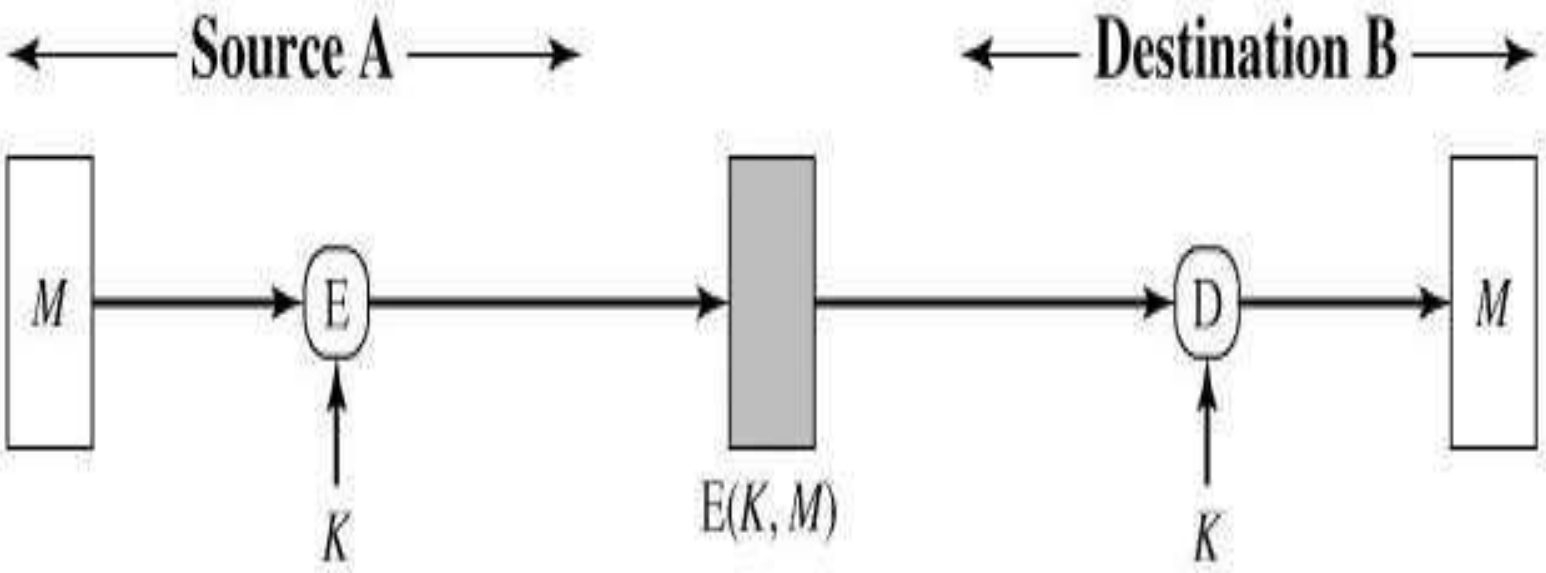
- Denial of receipt of message by destination.
- Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered.
- Message authentication may also verify sequencing and timeliness

AUTHENTICATION FUNCTIONS

- The types of functions that may be used to produce an authenticator may be grouped into three classes
- – Message encryption
- – Message authentication code (MAC)
- – Hash function:

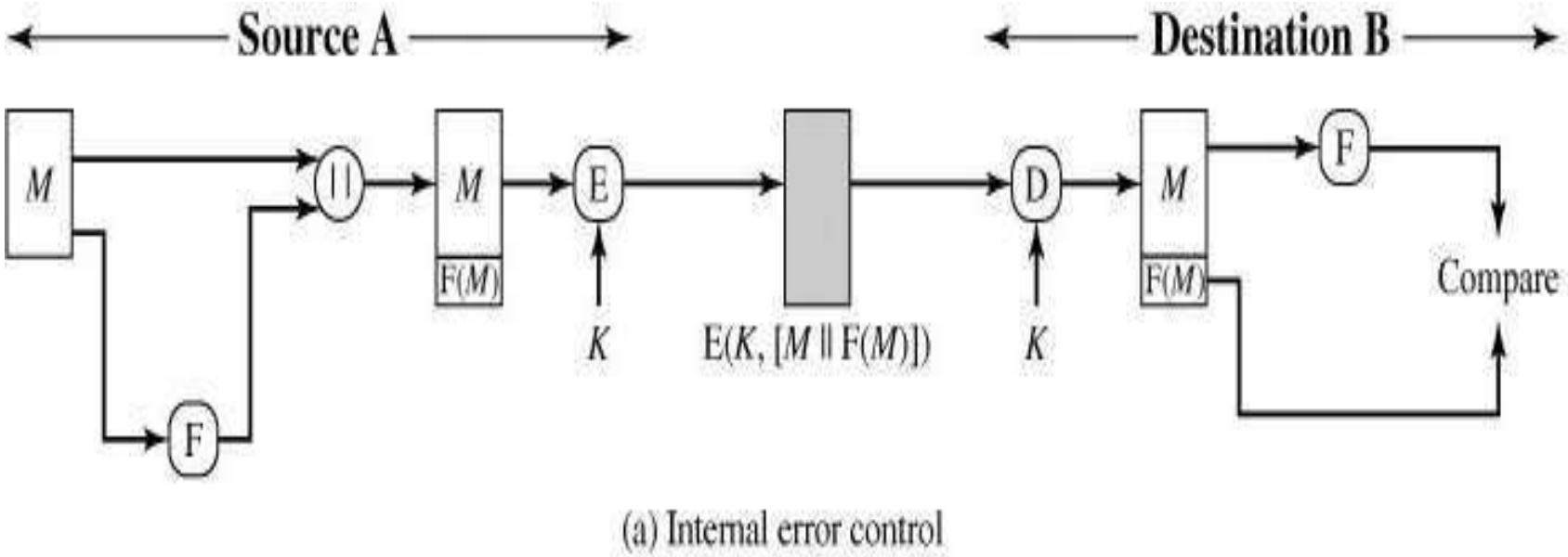
- **Message encryption –**
- The cipher text of the entire message serves as its authenticator.
- Message encryption by itself can provide a measure of authentication.
- The analysis differs for symmetric and public-key encryption schemes.
  - Symmetric Encryption
  - Public-Key Encryption: Confidentiality
  - Public-Key Encryption: Authentication
  - Public-Key Encryption: Confidentiality and authentication

Symmetric Encryption



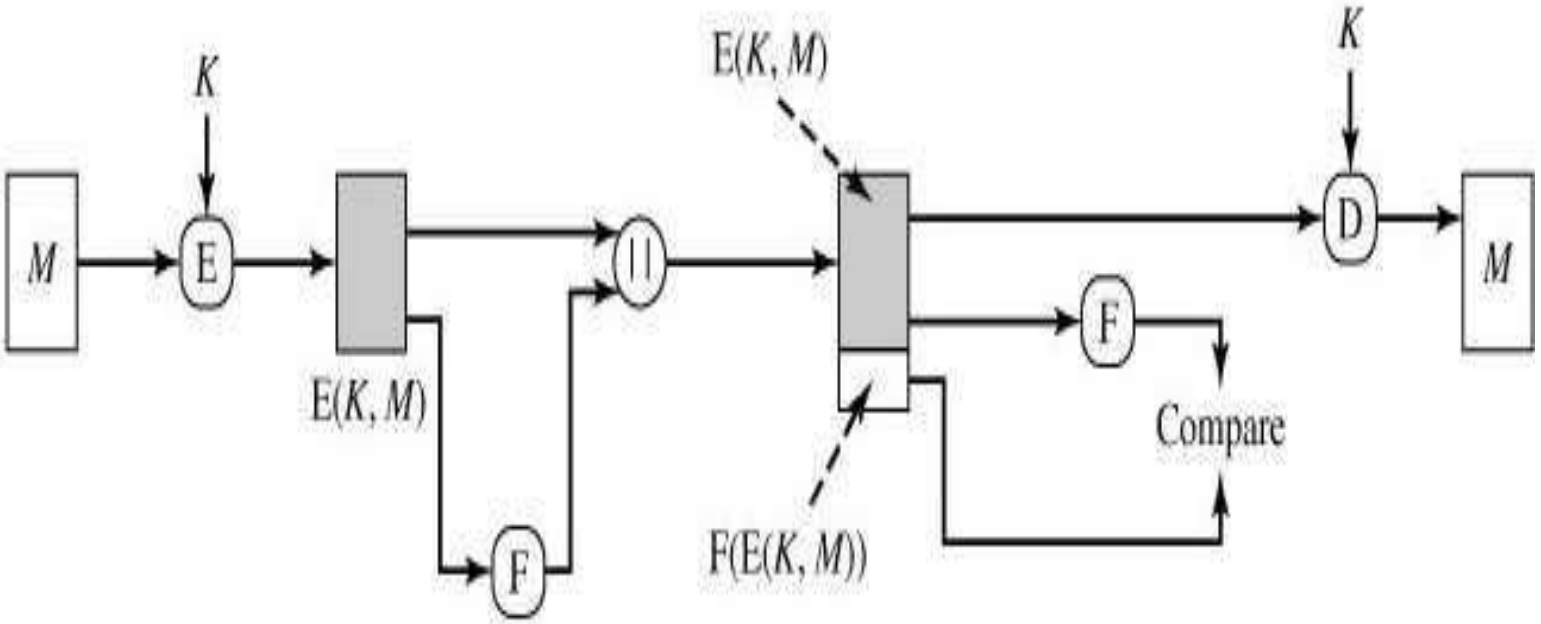
(a) Symmetric encryption: confidentiality and authentication

Internal Error Control - Append an error- detecting code, or frame check sequence (FCS) or checksum with the message



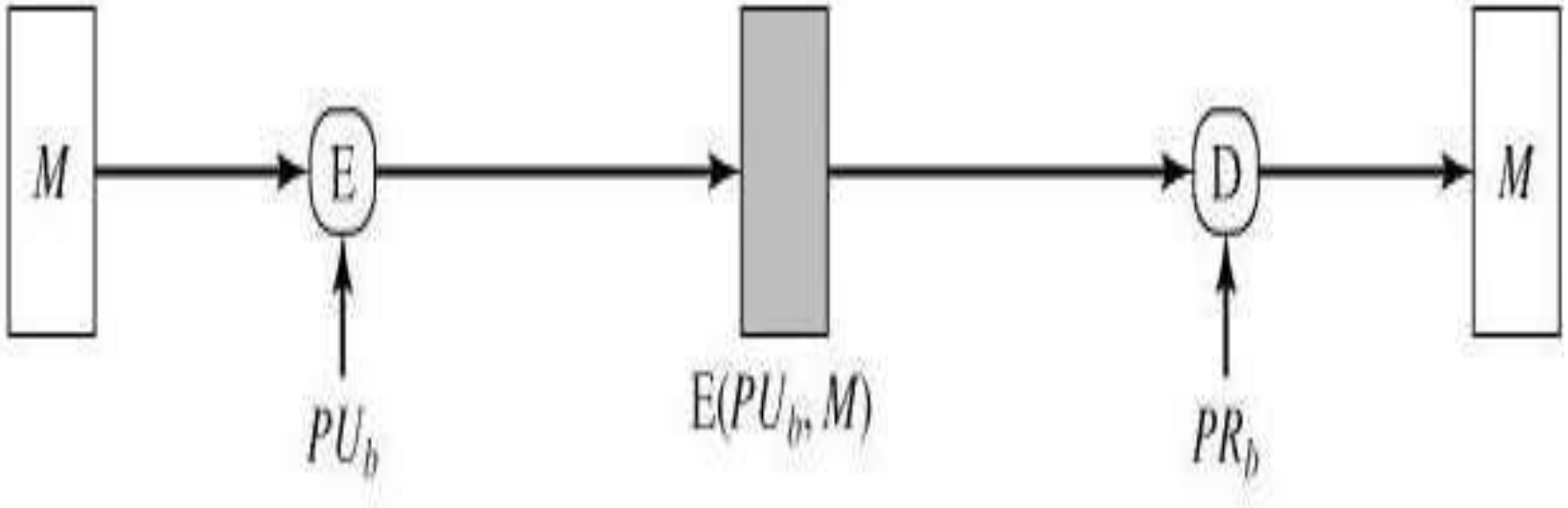
(a) Internal error control

External error control



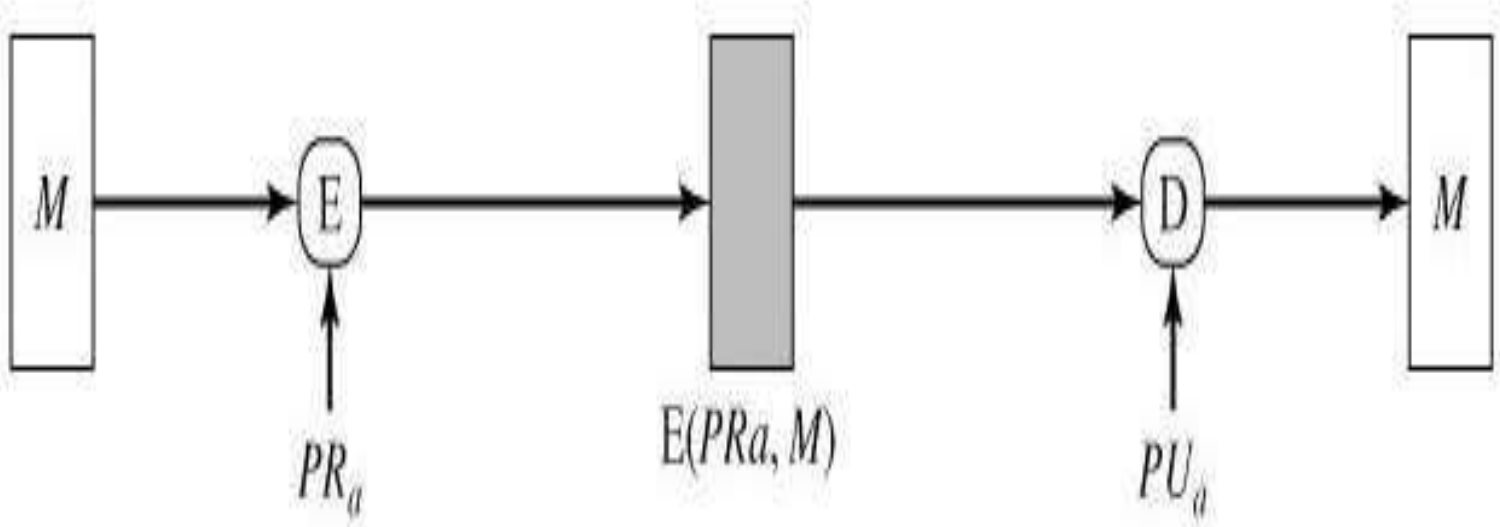
(b) External error control

2. Public-Key Encryption: Confidentiality



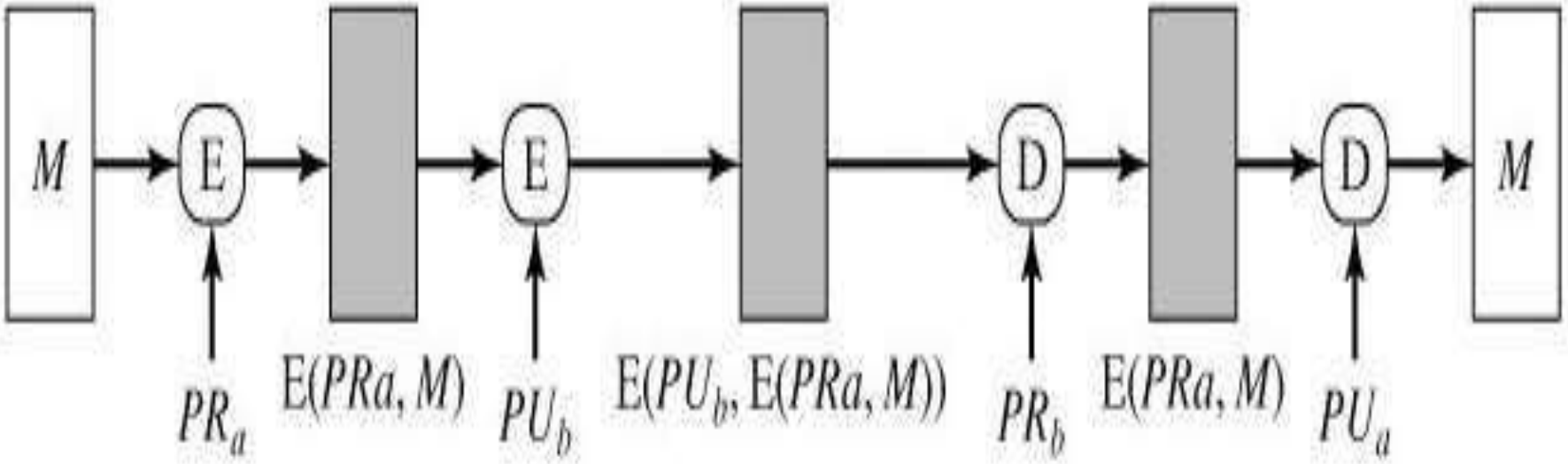
(b) Public-key encryption: confidentiality

3. Public-Key Encryption: Authentication



(c) Public-key encryption: authentication and signature

4. Public-Key Encryption: Confidentiality and authentication



(d) Public-key encryption: confidentiality, authentication, and signature

Message authentication code (1)

- An authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum or MAC** that is appended to the message.
- two communicating parties, say A and B, share a common secret key *K*.

Message authentication code (2)

- When A has a message to send to B, it calculates the MAC as a function of the message and the key:

$$\mathbf{MAC} = \mathbf{C}(\boldsymbol{K}, \boldsymbol{M})$$

Where ***M = input message***

C = MAC function

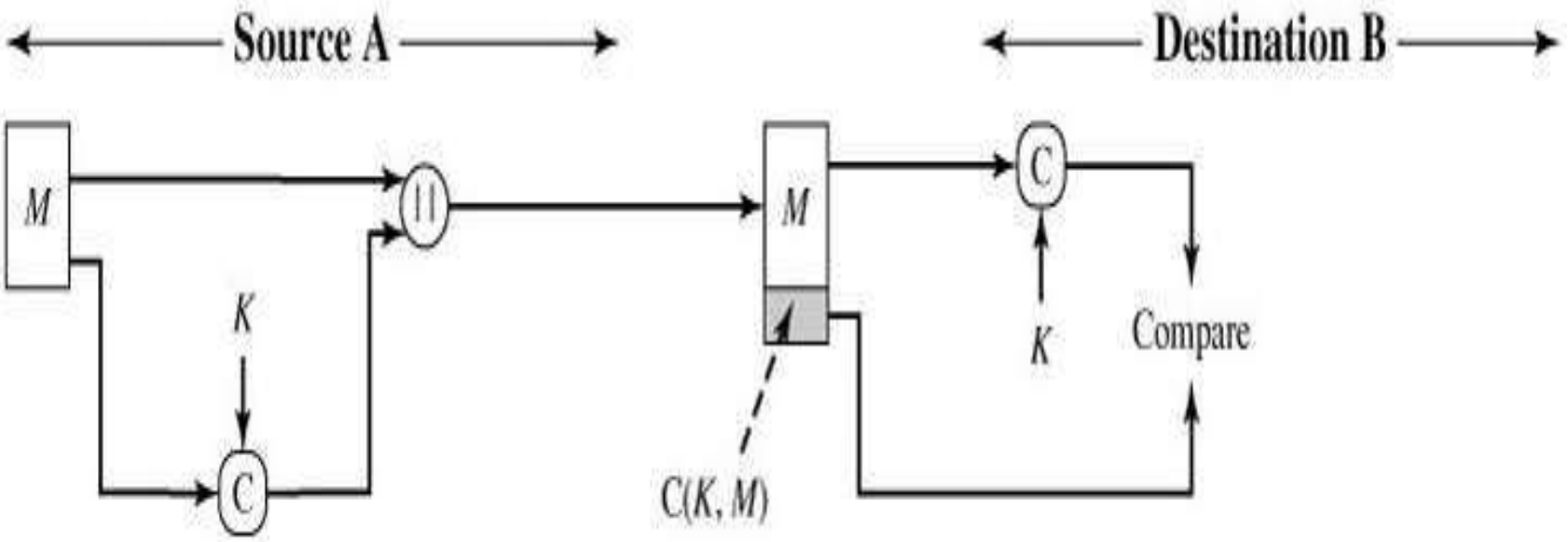
K = shared secret key

MAC = message authentication code

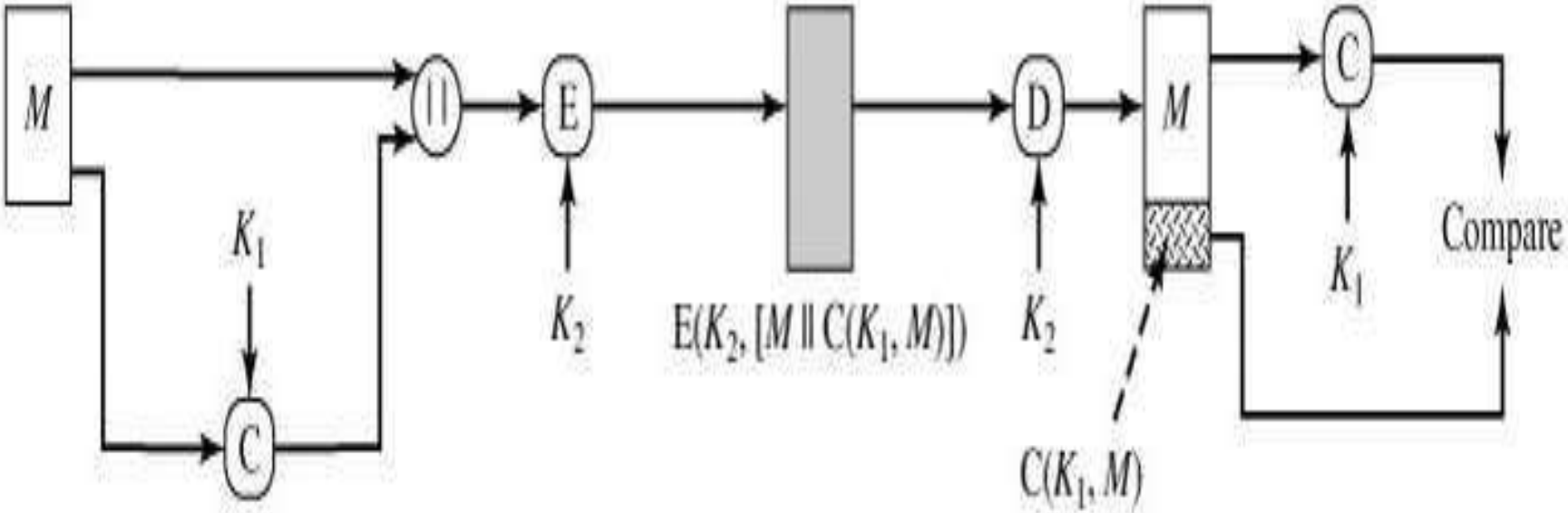
Message authentication code (3)

- Message plus MAC are transmitted to the intended recipient.
- recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC.
- received MAC is compared to the calculated MAC as shown in the figure

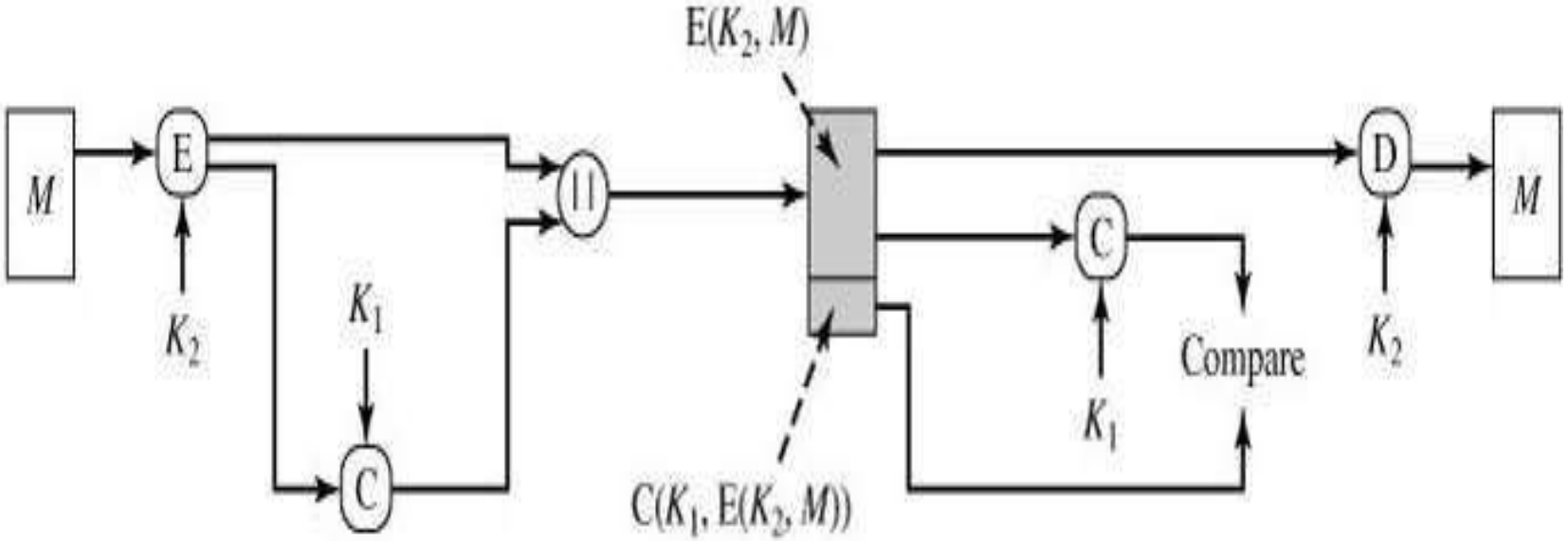
Message authentication code (4)



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

(iii) HASH FUNCTION

- A variation on the message authentication code is the one-way hash function.
- As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a **hash code H(M).**

- Unlike a MAC, a hash code does not use a key but is a function only of the input message.
- The hash code is also referred to as a **message digest or hash value.**
- The hash code is a function of all the bits of the message and provides an error-detection capability:
- A change to any bit or bits in the message results in a change to the hash code.

HASH FUNCTIONS

- A hash value h is generated by a function H of the form **h = H(M),** where

  M is a variable-length message

  H(M) is the fixed-length hash value.

- The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.
- The receiver authenticates that message by re computing the hash value.
- Because the hash function itself is not considered to be secret, some means is required to protect the hash value.

Requirements for a Hash Function(1)

- The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data.
- To be useful for message authentication, a hash function H must have the following properties:

Requirements for a Hash Function(2)

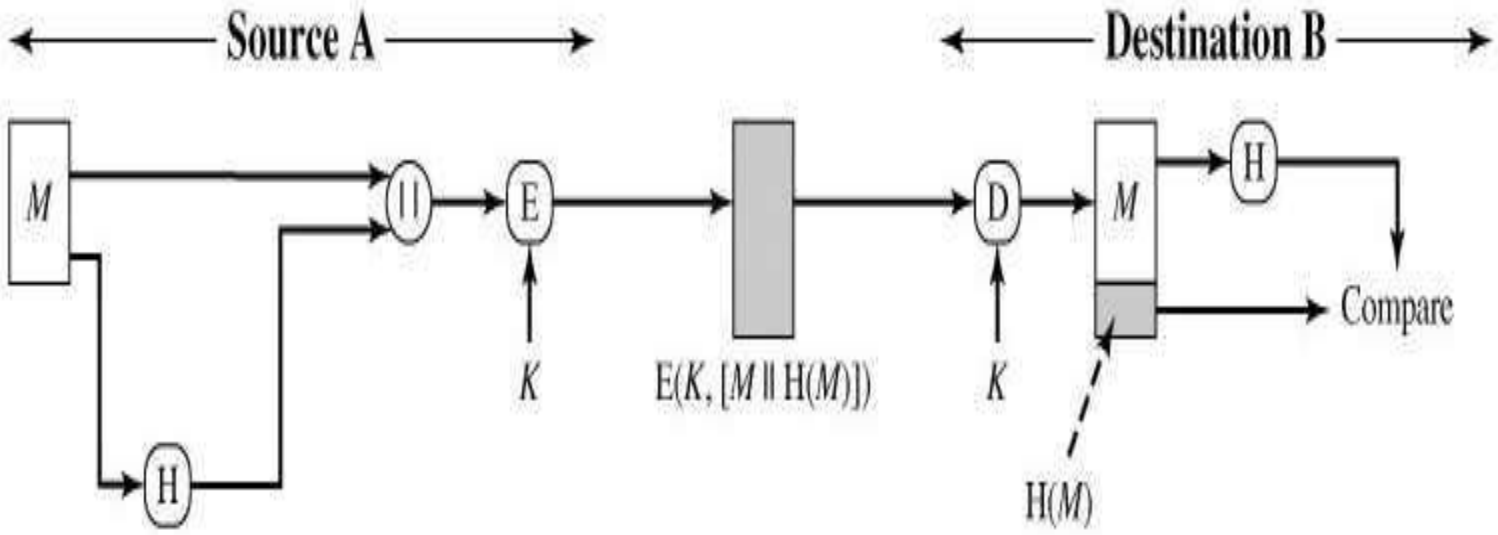
1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.

Requirements for a Hash Function(3)

- For any given block x, it is computationally infeasible to find y not equal to x such that H(y) = H(x). This is sometimes referred to as **weak collision resistance.**
- It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance.**
- The variety of ways in which a hash code can be used to provide message authentication, as follows:

a)

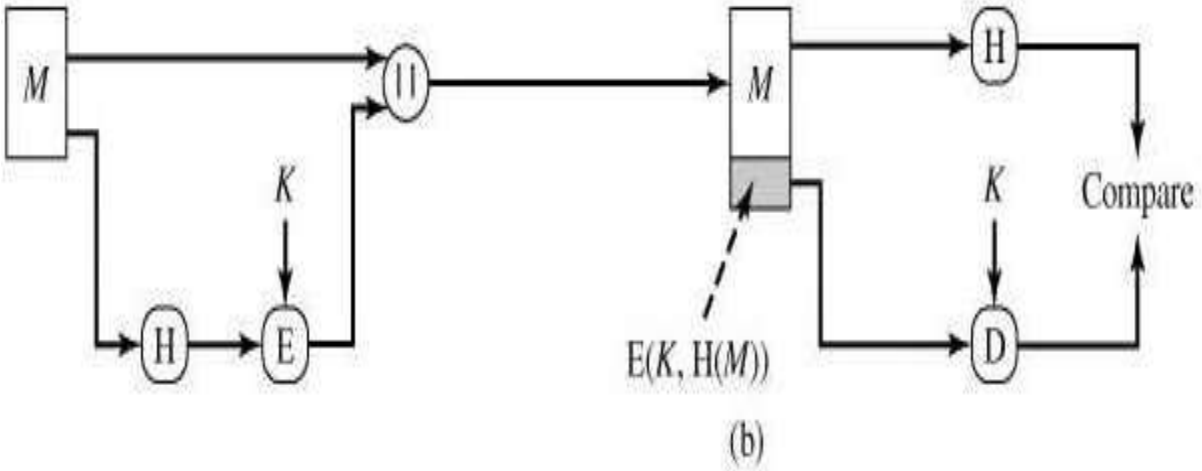
- The message plus concatenated hash code is encrypted using symmetric encryption.
- This is identical in structure to the internal error control strategy.
- The same line of reasoning applies: Because only A and B share the secret key, the message must have come from A and has not been altered.
- The hash code provides the structure or redundancy required to achieve authentication.
- Because encryption is applied to the entire message plus hash code, confidentiality is also provided.



(a)

b)

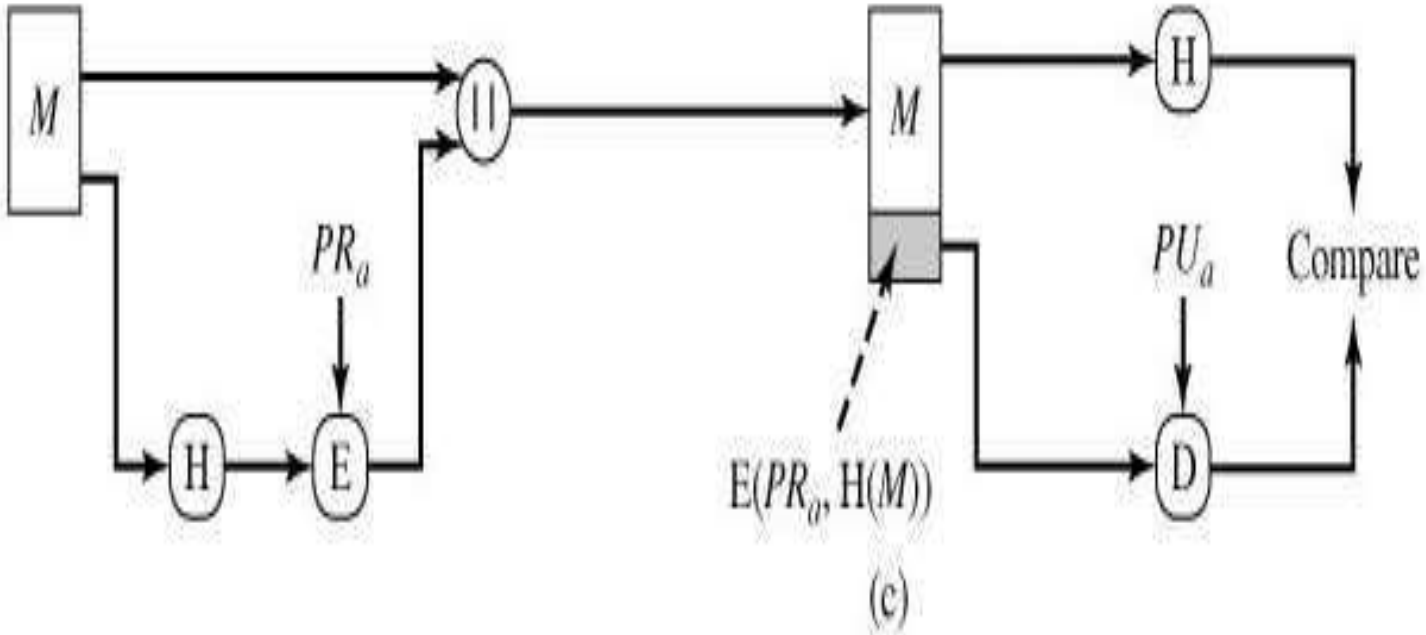
- Only the hash code is encrypted, using symmetric encryption.
- This reduces the processing burden for those applications that do not require confidentiality.



(b)

c)

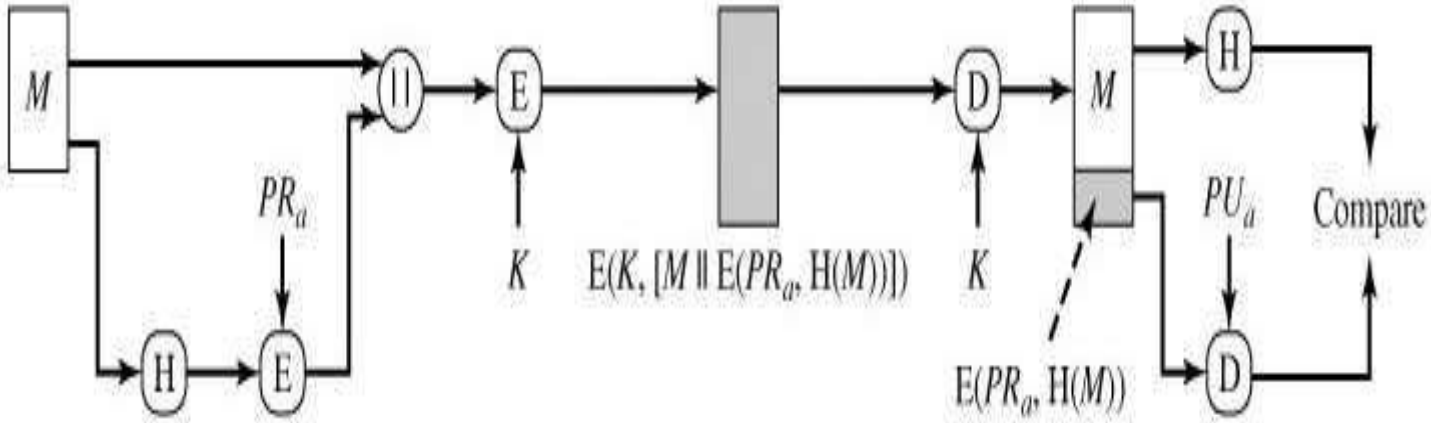
- Only the hash code is encrypted, using public-key encryption and using the sender's private key. As with (b), this provides authentication.
- It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.



(c)

d)

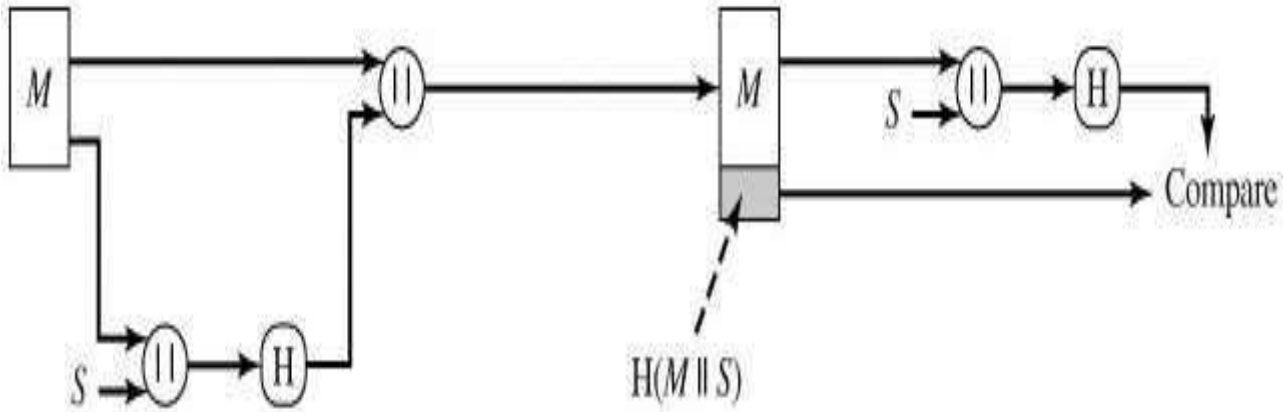
- If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.



(d)

e)

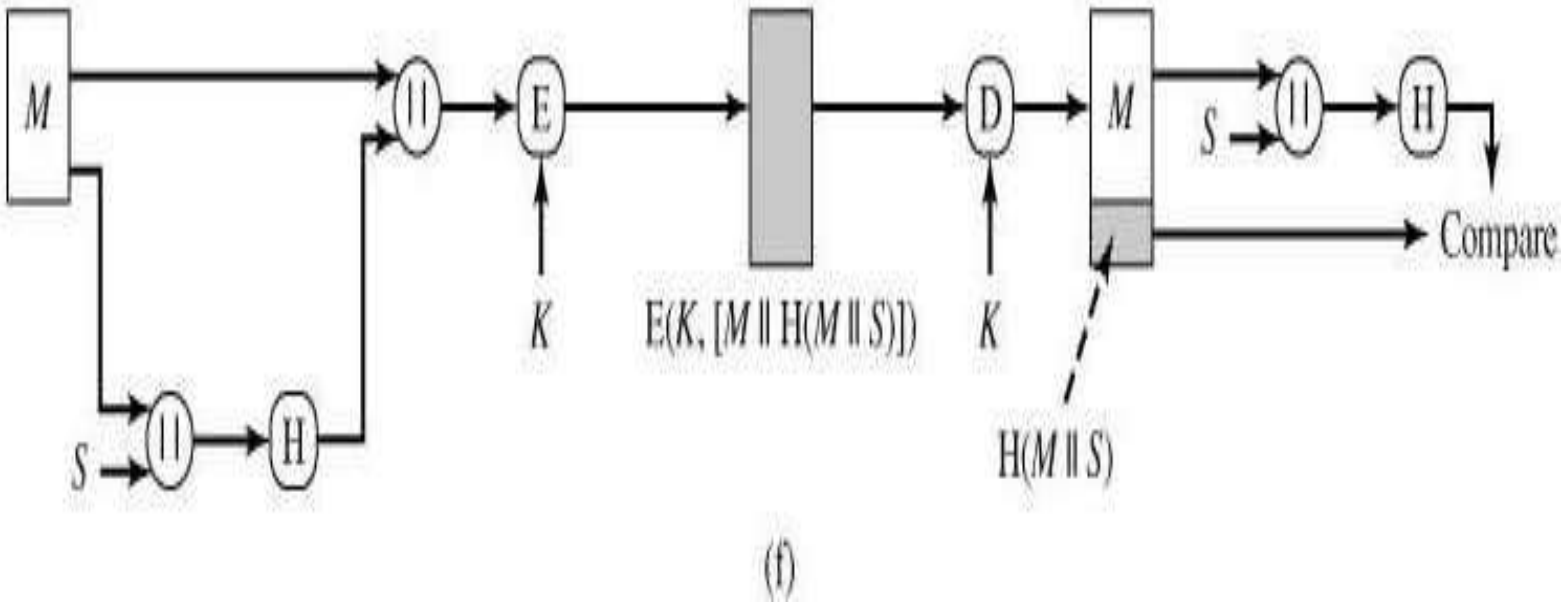
- It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the
- concatenation of M and S and appends the resulting hash value to M.
- Because B possesses S, it can re compute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.



(e)

f)

- Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.



(f)

Secure Hash Algorithm (SHA- 512)

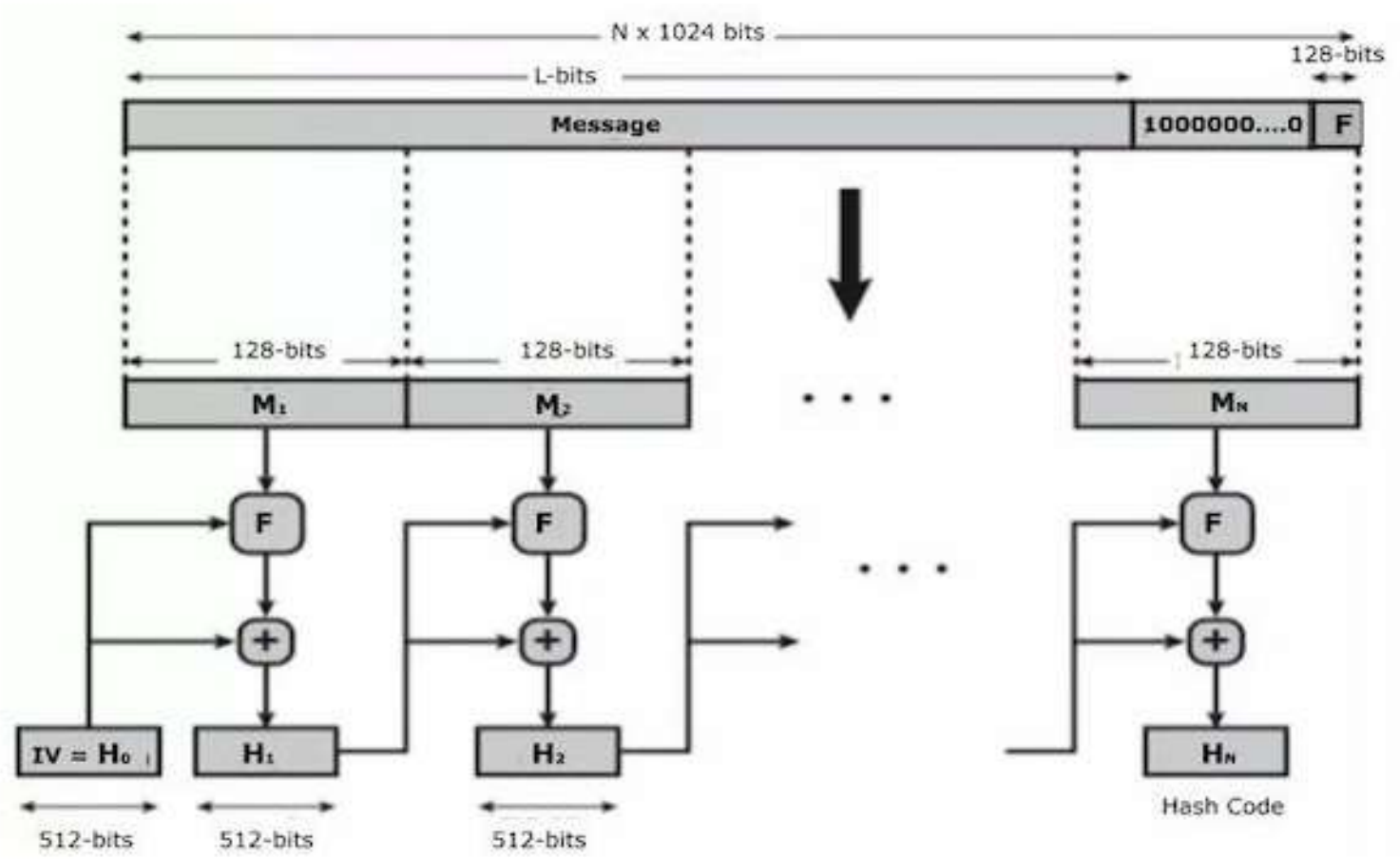
- SHA-512, or Secure Hash method 512, is a hashing technique that converts text of arbitrary length into a fixed-size string.
- Each output has a SHA-512 length of 512 bits (64 bytes).
- This algorithm is frequently used for email address hashing, password hashing, and digital record verification.

How SHA-512 Works?

- **Initialization** − It starts with eight hash values calculated from the square roots of the initial eight prime numbers.
- **Pre-processing** − The input message is padded so that it is a multiple of the Block size. The original message's 128-bit length (before padding) is added to the very end of the padded message.
- **Parsing** − The message is then separated into 1024-bit parts.
- **Main Loop** − The main loop analyses each 1024-bit block in 80 rounds, manipulating the data via logical operations, bitwise shifts, and modular arithmetic.
- **Output** − After all of the blocks have been processed, the resulting 512-bit message digest is output as the hash.

Algorithm

- **Message Padding** − First, your message is padded to ensure that it is the correct size for the algorithm. This ensures that it can be broken down into blocks and processed.
- **Initial hash values** − The algorithm starts with eight initial hash values. These set values serve as the basis for the hashing procedure.
- **Message processing** − The padded message is divided into blocks. Each block progresses over a series of stages known as rounds. In each round, the block is mixed and adjusted using specific techniques.
- **Final hash value** − After all blocks have been examined, the hash value is computed. This hash value serves as a unique fingerprint for the original message.
- **Output** − The SHA-512 algorithm generates the final hash result, which is generally a string of hexadecimal integers. This is the value returned after hashing your original message.


N x 1024 bits
128-bits
L-bits
Message
1000000....0
F
128-bits
128-bits
128-bits
M1
M2
MN
F
F
F
IV = H0
H1
H2
HN
Hash Code
512-bits
512-bits
512-bits

Message authentication codes: HMAC, CMAC, Digital signatures

- What is HMAC?
- HMAC (short for "Keyed-Hash Message Authentication Code") is a cryptographic hash function that uses a secret key as input to the hash function along with the message being hashed. The resulting hash value is unique to the message and the secret key, and can be used to verify the integrity and authenticity of the message.
- HMAC is widely used as a secure way to authenticate messages in various communication protocols, including HTTP, SSL, SSH, and many others.

How HMAC Works?

- A message is input to the HMAC function along with a secret key.
- The HMAC function applies a cryptographic hash function to the message and the secret key, generating a unique hash value.
- The hash value is output by the HMAC function and can be used to authenticate the message.

How to Implement HMAC?

- Choose a cryptographic hash function to use for the HMAC. Common choices include SHA-1, SHA-256, and SHA-512.
- Choose a secret key to use for the HMAC. The secret key should be a random, unique value that is kept secret from anyone who is not authorized to access the message.
- Input the message and the secret key to the HMAC function along with the desired length of the hash output and the specific hash algorithm to be used.
- The HMAC function will generate a unique hash value based on the message, the secret key, and the additional information.
- The hash value can then be output by the HMAC function and used to authenticate the message.

- CMAC (Cipher-based Message Authentication Code): A cryptographic technique for message authentication using a symmetric key block cipher algorithm such as AES.
- It provides integrity and authenticity for messages or data by producing a fixed-size output, known as a tag or MAC, which is sent along with the message.
- The receiver then computes the CMAC using the same secret key to verify the integrity of the message. CMAC is resistant to forgery and is widely used in various security protocols and applications.

- Digital signatures are the public key equivalent of private key message authentication codes (MACs).
- Whereas MACs use private keys to enable a message recipient to verify that a message has not been altered during transmission, signatures use a private/public key pair.
- The sender creates a message hash, signs it, and sends both the signature and the (unencrypted) message. The recipient calculates a hash over the message, decrypts the signature, and compares the decrypted signature to the hash value. If they match, the recipient can be fairly certain that the message did, in fact, come from the sender and was not altered during transmission.

AUTHENTICATION APPLICATIONS

Kerberos

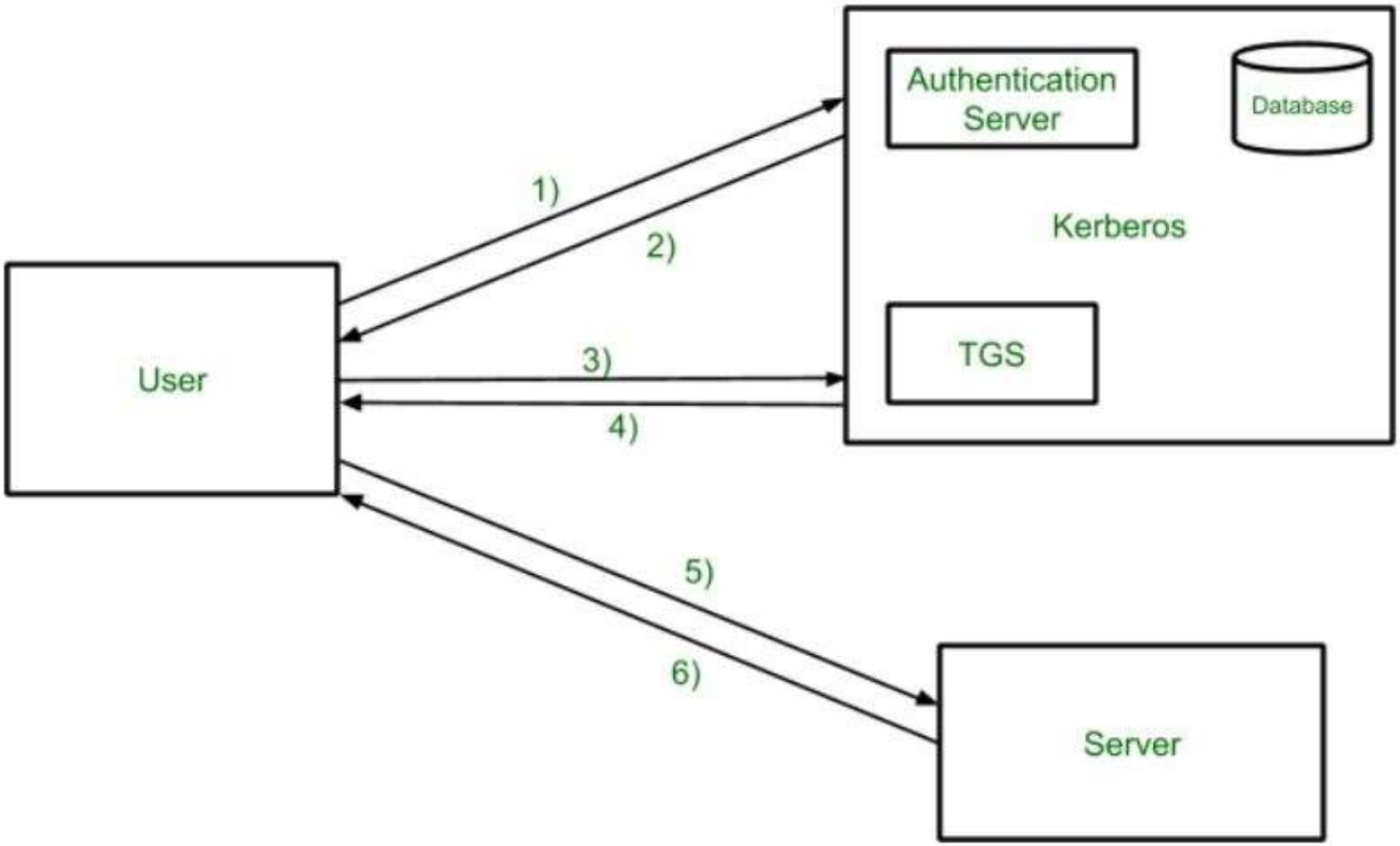
- **Kerberos** provides a centralized authentication server whose function is to authenticate users to servers and servers to users.
- In Kerberos Authentication server and database is used for client authentication.
- Kerberos runs as a third-party trusted server known as the Key Distribution Center (KDC). Each user and service on the network is a principal.

The main components of Kerberos are:

- **Authentication Server (AS):** The Authentication Server performs the initial authentication and ticket for Ticket Granting Service.
- **Database:** The Authentication Server verifies the access rights of users in the database.
- **Ticket Granting Server (TGS):** The Ticket Granting Server issues the ticket for the Server

Working of Kerberos

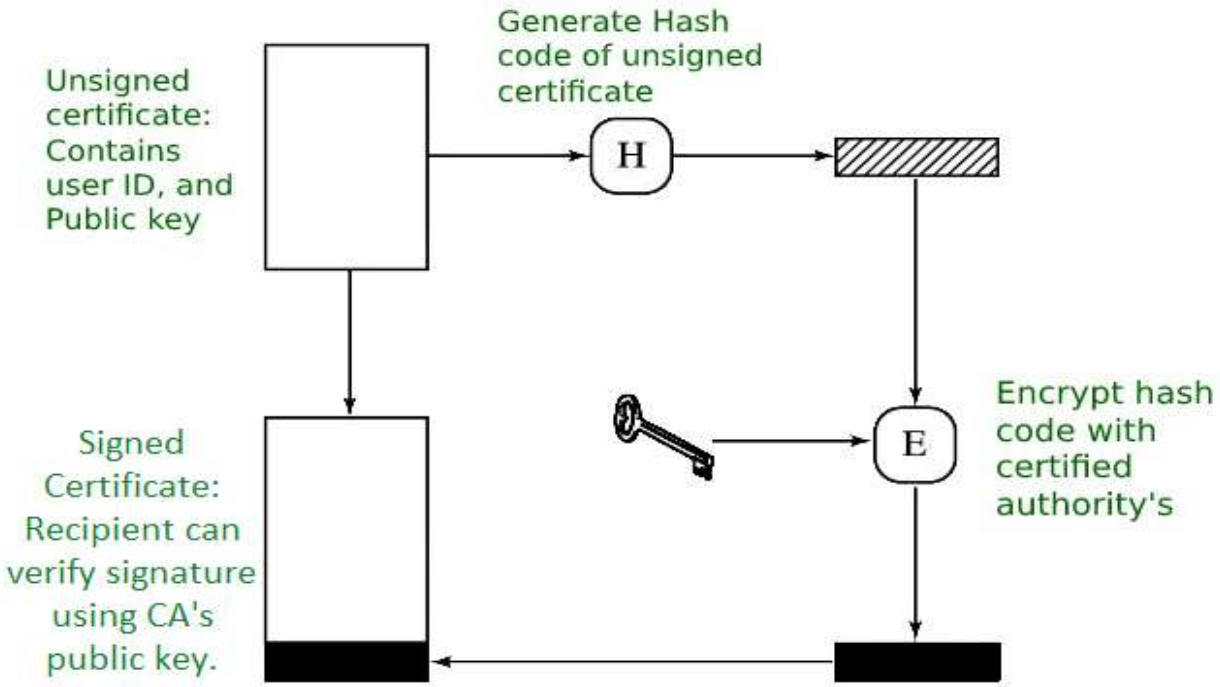
- **Step-1:** User login and request services on the host. Thus user requests for ticket-granting service.
  Step-2: Authentication Server verifies user's access right using database and then gives ticket-granting-ticket and session key. Results are encrypted using the Password of the user.
  Step-3: The decryption of the message is done using the password then send the ticket to Ticket Granting Server. The Ticket contains authenticators like user names and network addresses.
  Step-4: Ticket Granting Server decrypts the ticket sent by User and authenticator verifies the request then creates the ticket for requesting services from the Server.
  Step-5: The user sends the Ticket and Authenticator to the Server.
  Step-6: The server verifies the Ticket and authenticators then generate access to the service. After this User can access the services.


Authentication Server
Database
Kerberos
TGS
User
1)
2)
3)
4)
5)
6)
Server

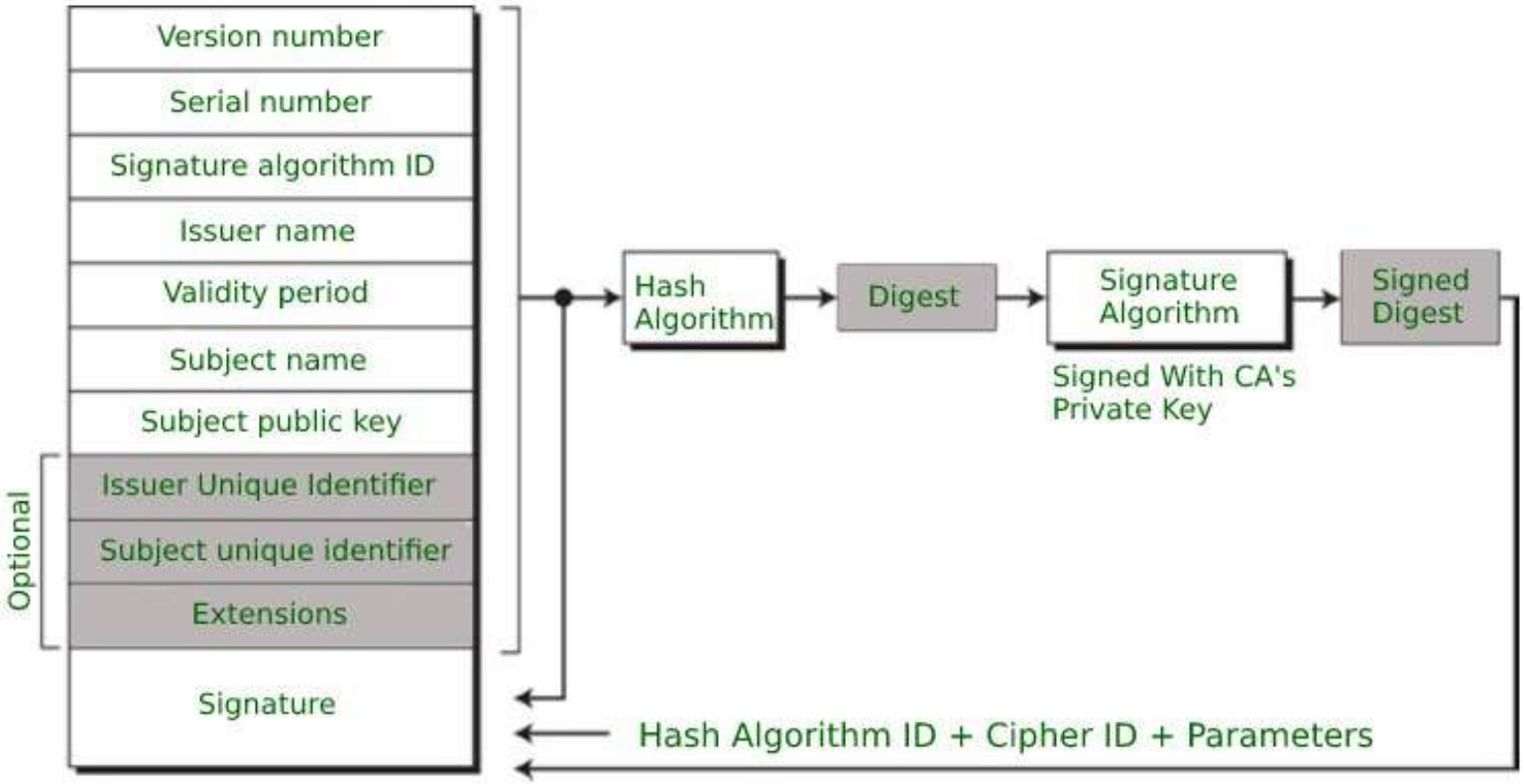
X.509 Authentication

- X.509 is a digital certificate that is built on top of a widely trusted standard known as ITU or International Telecommunication Union X.509 standard.
- X.509 digital certificate is a certificate-based authentication security framework that can be used for providing secure transaction processing and private information.
- These are primarily used for handling the security and identity in computer networking and internet-based communications.

Working of X.509 Authentication Service Certificate:



Format of X.509 Authentication Service Certificate:



Web Security:

Web security considerations, Secure Socket Layer, and Transport Layer Security, Secure Electronic Transaction.

E-Mail Security:

Pretty Good Privacy, S/MIME.

Objectives of Web Security

- To identify variety of threats and web security issues
- To identify need for security
- Keep security simple
- Detect intrusions with the help of security mechanism

Web Security Considerations

- World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets.
- The Web presents new challenges not generally appreciated in the context of computer and network security:
- The Internet is two way. Unlike traditional publishing environments, even electronic publishing systems involving Teledex, voice response or fax-back, the Web is vulnerable to attacks on the Web servers over the Internet.

Web Security Threats

A Comparison of Threats on the Web

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

Web Traffic Security Approaches

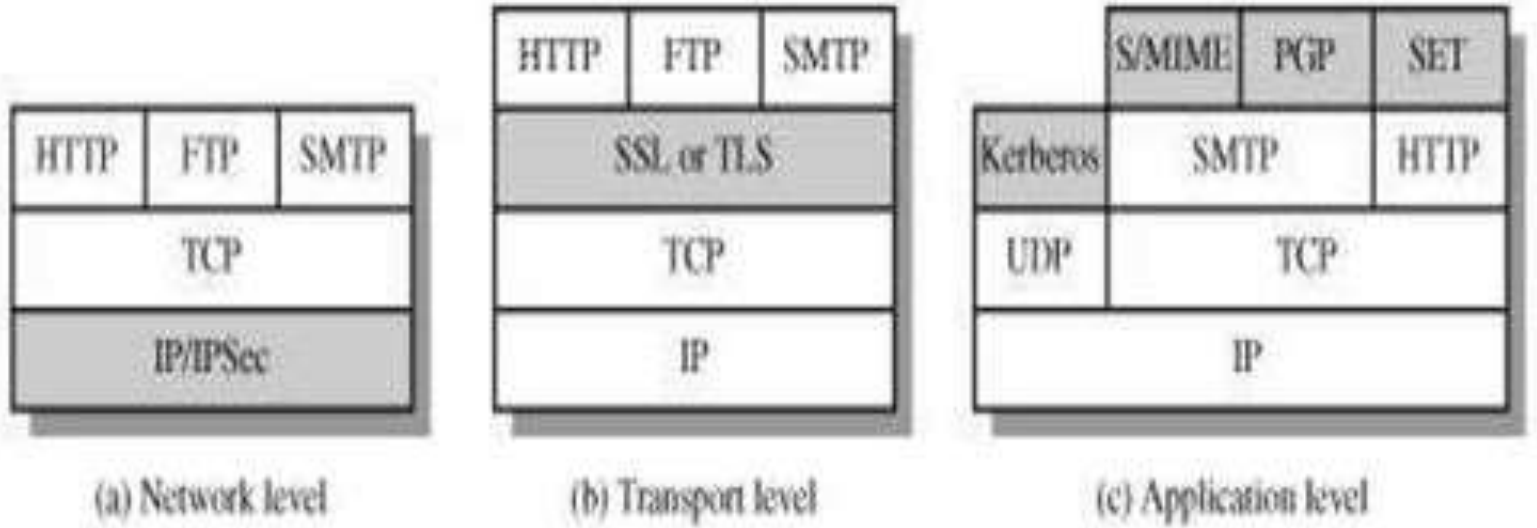


Figure 1.1 Relative Location of Security Facilities in the TCP/IP Protocol Stack

Secure Sockets Layer

- SSL was developed by Netscape to provide security when transmitting information on the Internet.
- The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP).
- SSL provides for secure communication between client and server by allowing mutual

SSL Architecture

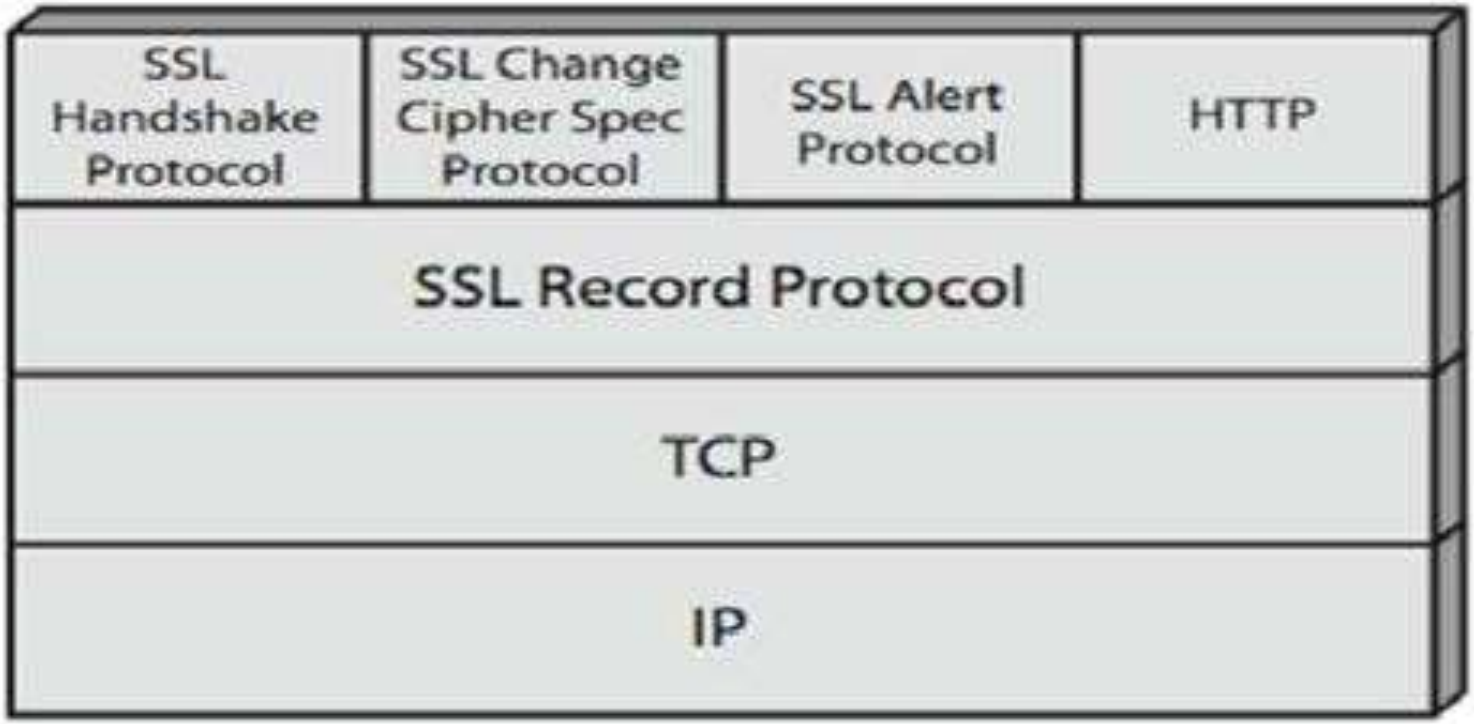


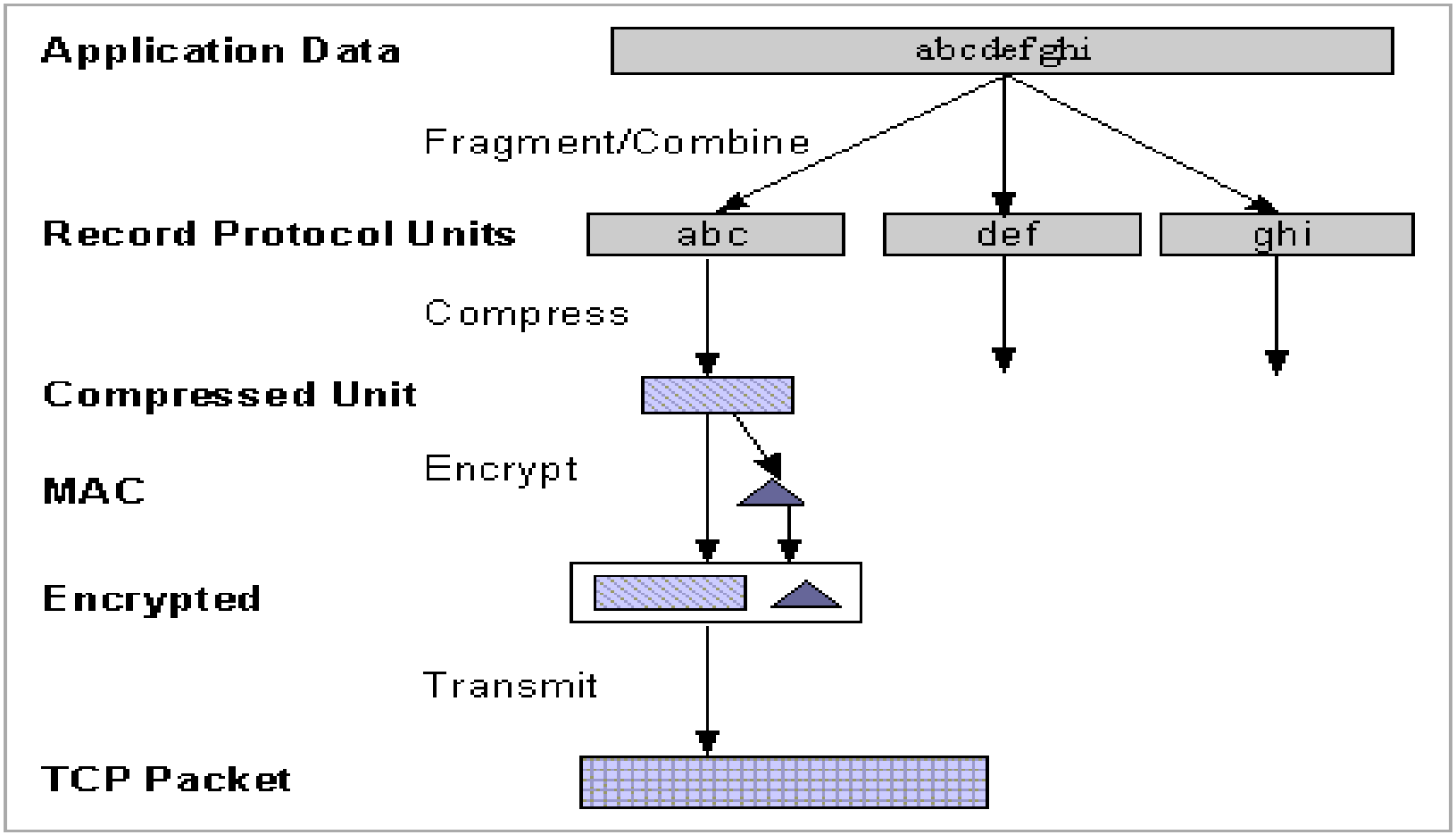
Figure 2.1 SSL Protocol Stack

- Two important SSL concepts are the SSL connection and the SSL session:

- • **Connection:** A connection is a network transport that provides a suitable type of service, such connections are transient, peer-to-peer relationships, associated with one session.
-
- • **Session:** An SSL session is an association between a client and a server, created by the Handshake Protocol. Sessions define a set of cryptographic security parameters, which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

SSL Record Protocol

- SSL Record Protocol defines two services for SSL connections:
- **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC), which is similar to HMAC.
- **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported.


Application Data
abcdefghi
Fragment/Combine
Record Protocol Units
abc
def
ghi
Compress
Compressed Unit
Encrypt
MAC
Encrypted
Transmit
TCP Packet

SSL record format

- **Content Type (8 bits):** The higher layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment.

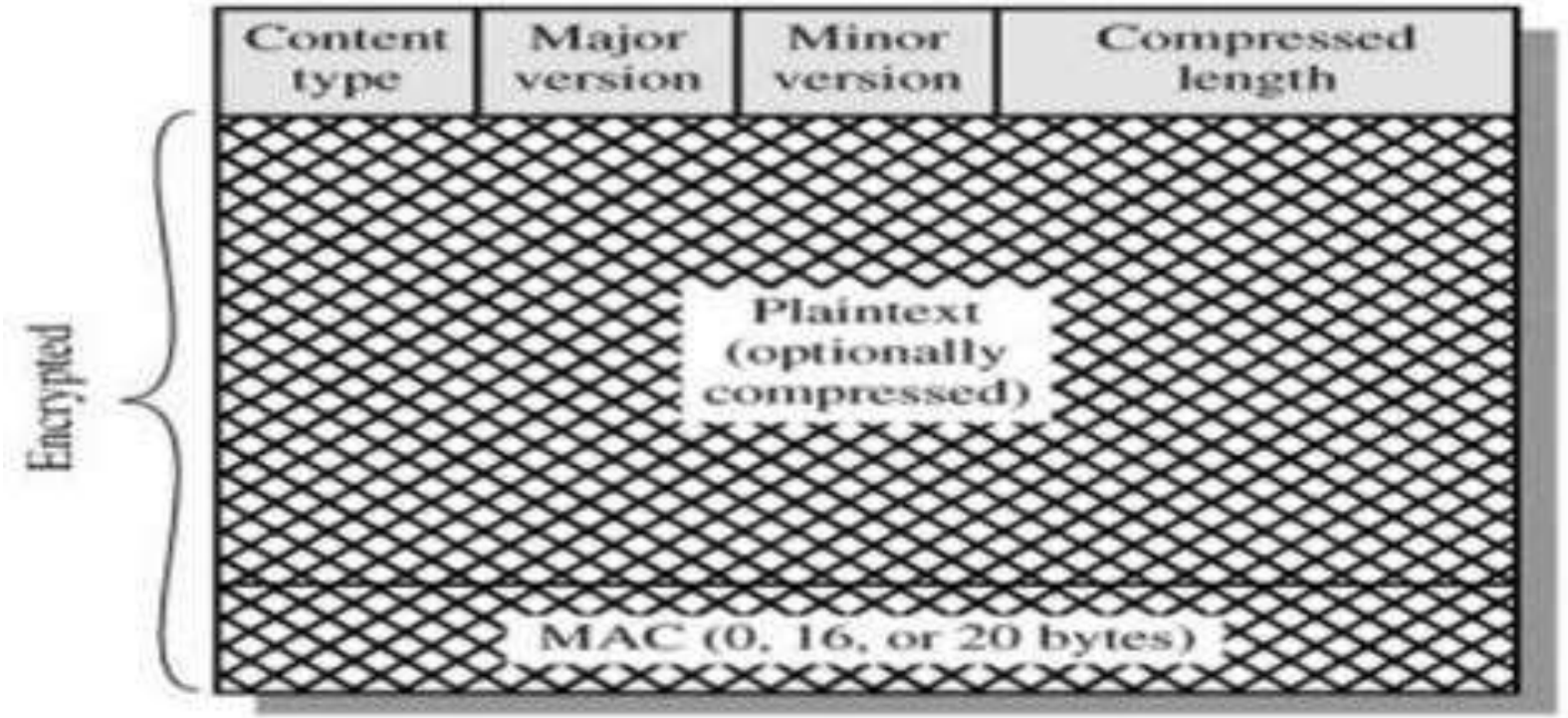


Figure 2.3 SSL Record Format

SSL Change Cipher Spec Protocol

- The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use the SSL Record Protocol, and it is the simplest.
- This protocol consists of a single message, which consists of a single byte with the value 1.
- The purpose of this message is to cause the pending state to be copied into the current state, which updates the cipher suite to be used on this connection.

SSL Alert Protocol

- The Alert Protocol is used to convey SSL-related alerts to the peer entity.
- As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.
- Each message in this protocol consists of two bytes. The first byte takes the value **warning(1)** or **fatal(2)** to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this session may be established. The second byte contains a code that indicates the specific alert.

The fatal alerts are listed below

- Unexpected_message
- Bad_record_mac
- Decompression_failure:
- Handshake_failure
- illegal_paramete

The remainder of the alerts

- close_notify
- no_certificate
- bad_certificate
- unsupported_certificate
- certificate_expired
- certificate_unknown

SSL Handshake Protocol

SSL Handshake protocol ensures establishment of reliable and secure session between client and server and also allows server & client to:

- Authenticate each other
- To negotiate encryption & MAC algorithms
- To negotiate cryptographic keys to be used

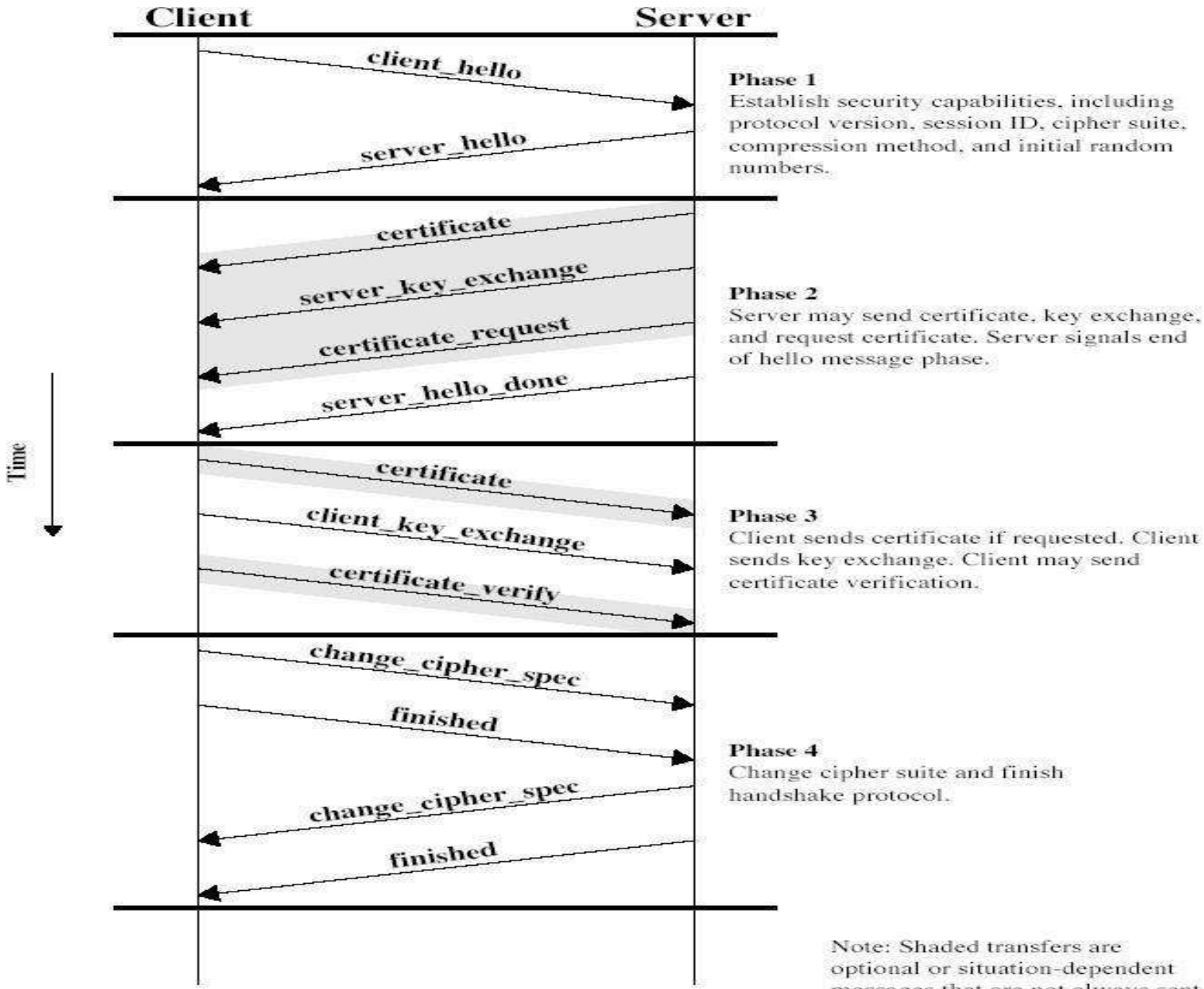
- The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown below and each message has three fields:
- • Type (1 byte): Indicates one of 10 messages.
- • Length (3 bytes): The length of the message in bytes.
- • Content (>=0 bytes): The parameters associated with this message



(c) Handshake Protocol

- The following shows the initial exchange needed to establish a logical connection between client and server.
- The exchange can be viewed as having four phases.in phases

1. Establish Security Capabilities
2. Server Authentication and Key Exchange
3. Client Authentication and Key Exchange
4. Finish


Client
Server
client_hello
server_hello
Phase 1
Establish security capabilities, including protocol version, session ID, cipher suite, compression method, and initial random numbers.
certificate
server_key_exchange
certificate_request
server_hello_done
Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.
Time
certificate
client_key_exchange
certificate_verify
Phase 3
Client sends certificate if requested. Client sends key exchange. Client may send certificate verification.
change_cipher_spec
finished
change_cipher_spec
finished
Phase 4
Change cipher suite and finish handshake protocol.
Note: Shaded transfers are optional or situation-dependent messages that are not always sent.

TRANSPORT LAYER SECURITY

- TLS was released in response to the Internet community's demands for a standardized protocol.
- TLS (Transport Layer Security), defined in RFC 2246, is a protocol for establishing a secure connection between a client and a server.
- TLS (Transport Layer Security) is capable of authenticating both the client and the server and creating a encrypted connection between the two.
- Many protocols use TLS (Transport Layer Security) to establish secure connections, including HTTP, IMAP, POP3, and SMTP. The TLS Handshake Protocol first negotiates key exchange using an asymmetric algorithm such as RSA or Diffie-Hellman.

- The TLS Record Protocol then begins opens an encrypted channel using a symmetric algorithm such as RC4, DES, or 3DES.
- The TLS Record Protocol is also responsible for ensuring that the communications are not altered in transit.
- Hashing algorithms such as SHA are used for this purpose.
- RFC 2246 is very similar to SSLv3. There are some minor differences ranging from protocol version numbers to generation of key material.

- **Version Number:** The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 1.

- **Message Authentication Code:** Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length.
- **Pseudorandom Function:** TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation.

SET (SECURE ELECTRONIC TRANSACTION)

- SET is an open encryption and security specification designed to protect credit card transactions on the Internet.
- SET is not itself a payment system. Rather it is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network, such as the Internet, in a secure fashion. In essence,

SET provides three services:

- Provides a secure communications channel among all parties involved in a transaction
- Provides trust by the use of X.509v3 digital certificates
- Ensures privacy because the information is only available to parties in a transaction when and where necessary

SET Requirements

- Provide confidentiality of payment and ordering information Ensure the integrity of all transmitted data
- Provide authentication that a cardholder is a legitimate user of a credit card account Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution
- Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction
- Create a protocol that neither depends on transport security mechanisms nor prevents their use
- Facilitate and encourage interoperability among software and network providers

SET Key Features

To meet the requirements, SET incorporates the following features:

- Confidentiality of information
- Integrity of data
- Cardholder account authentication
- Merchant authentication

SET Participants

- **Cardholder:** purchasers interact with merchants from personal computers over the Internet
- **Merchant:** a person or organization that has goods or services to sell to the cardholder
- **Issuer:** a financial institution, such as a bank, that provides the cardholder with the payment card.
- **Acquirer:** a financial institution that establishes an account with a merchant and processes payment card authorizations and payments
- **Payment gateway:** a function operated by the acquirer or a designated third party that processes merchant payment messages
- **Certification authority (CA):** an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways

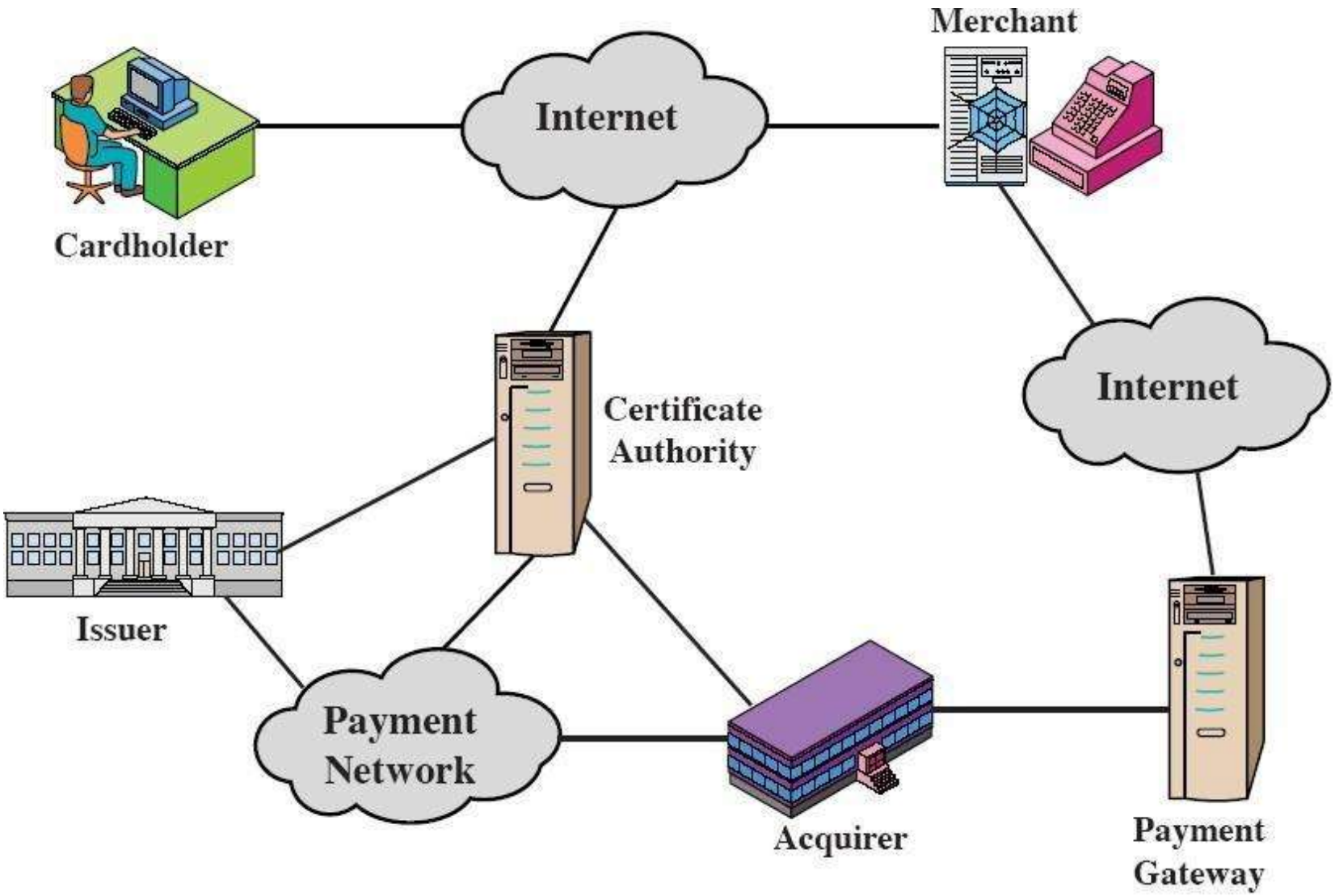


Figure 17.8 Secure Electronic Commerce Components

Events in a transaction

1. The customer obtains a credit card account with a bank that supports electronic payment and SET
2. The customer receives a X.509v3 digital certificate signed by the bank.
3. Merchants have their own certificates
4. The customer places an order
5. The merchant sends a copy of its certificate so that the customer can verify that it's a valid store
6. The order and payment are sent
7. The merchant requests payment authorization
8. The merchant confirms the order
9. The merchant ships the goods or provides the service to the customer
10. The merchant requests payment

E-Mail Security:

- Pretty Good Privacy
- The Pretty Good Privacy (PGP) secure email program .
- PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.
- It is independent of government organizations and runs on a wide range of systems, in both free & commercial versions.

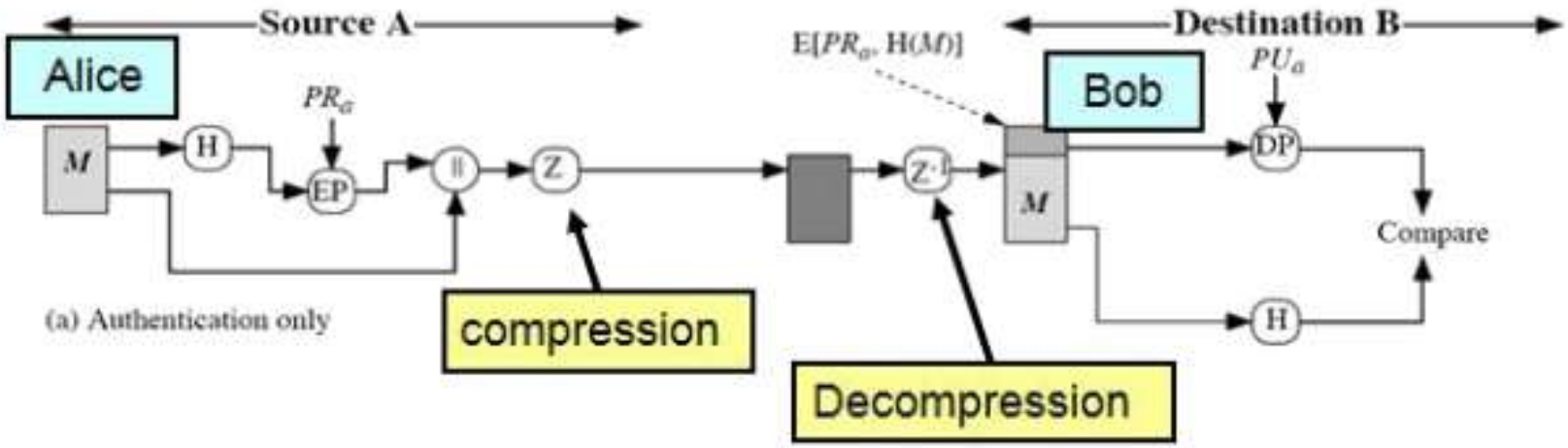
There are five important services in PGP

- Authentication (Sign/Verify)
- Confidentiality (Encryption/Decryption)
- Compression
- Email compatibility
- Segmentation and Reassembly

PGP Notations:

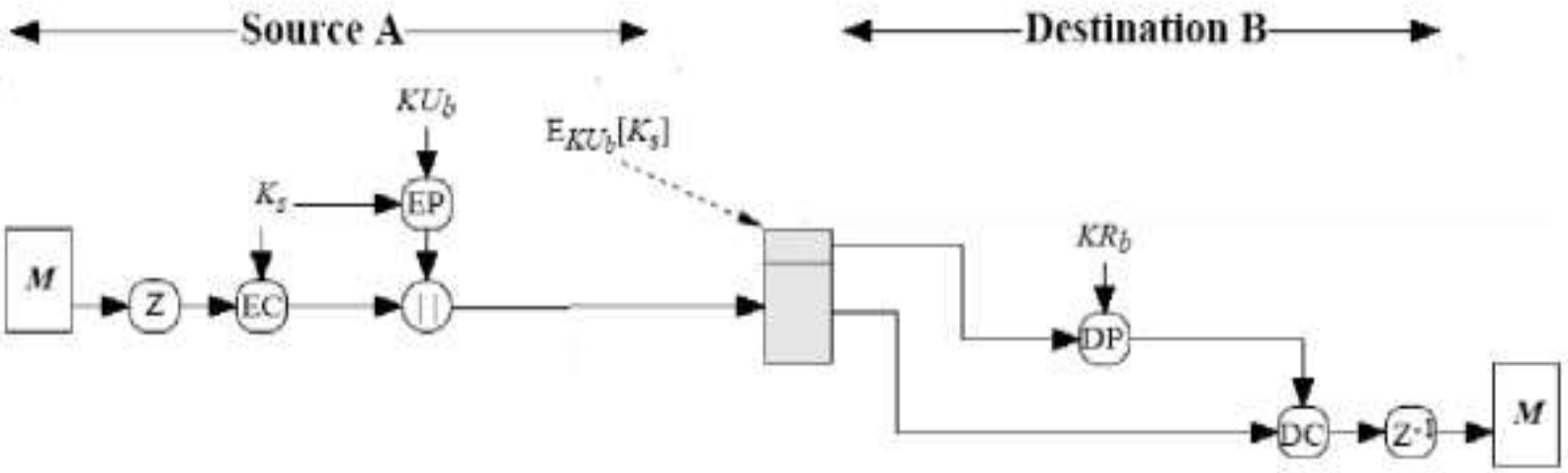
- Ks =session key used in symmetric encryption scheme
- PRa =private key of user A, used in public-key encryption scheme
- PUa =public key of user A, used in public-key encryption scheme
- EP = public-key encryption
- DP = public-key decryption
- EC = symmetric encryption
- DC = symmetric decryption
- H = hash function
- || = concatenation
- Z = compression using ZIP algorithm R64 = conversion to radix 64 ASCII format

PGP Operation- Authentication



1. sender creates message
2. use SHA-1 to generate 160-bit hash of message
3. signed hash with RSA using sender's private key, and is attached to message
4. Receiver uses RSA with sender's public key to decrypt and recover hash code
5. Receiver verifies received message using hash of it and compares with decrypted hash code

PGP Operation- Confidentiality



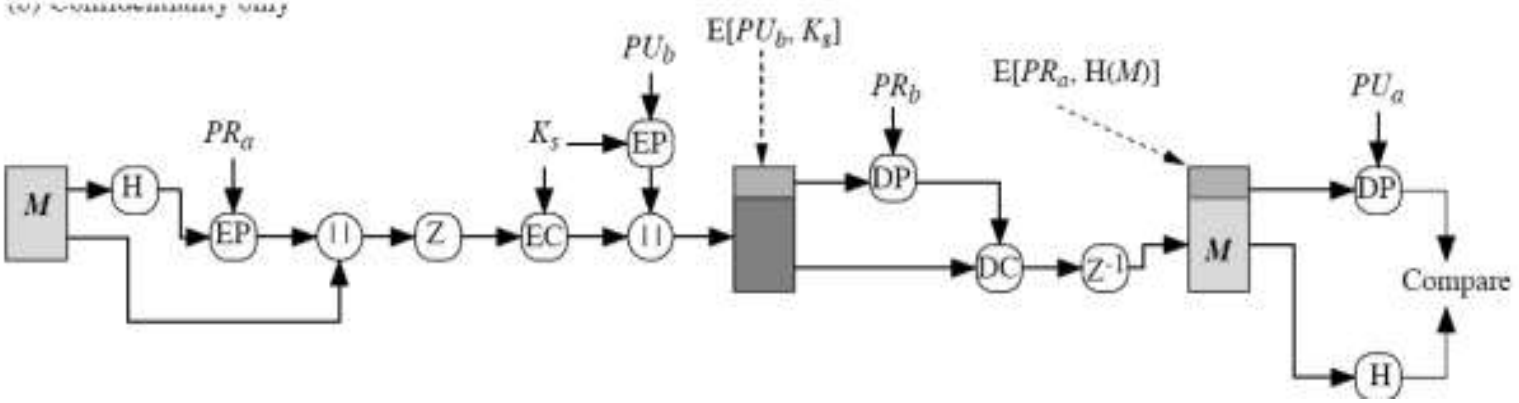
Sender:

1. Generates message and a random number (session key) only for this message
2. Encrypts message with the session key using AES, 3DES
3. Encrypts session key itself with recipient's public key using RSA
4. Attaches it to message

Receiver:

1. Recovers session key by decrypting using his private key
2. Decrypts message using the session key

PGP Operation – Confidentiality & Authentication



PGP Operation – Compression

- As a default, PGP compresses the message after applying the signature but before encryption.
- This has the benefit of saving space both for e-mail transmission and for file storage.
- The placement of the compression algorithm, indicated by Z for compression and Z1 for decompression is critical. The compression algorithm used is ZIP.

The signature is generated before compression for two reasons:

1. so that one can store only the uncompressed message together with signature for later verification
2. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm as the PGP compression algorithm is not deterministic

- Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

PGP Operation

- Email Compatibility When PGP is used, at least part of the block to be transmitted is encrypted, and thus consists of a stream of arbitrary 8-bit octets.
- However many electronic mail systems only permit the use of ASCII text.
- To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters.
- It uses radix-64 conversion, in which each group of three octets of binary data is mapped into four ASCII characters.
- This format also appends a CRC to detect transmission errors. The use of radix 64 expands a message by 33%, but still an overall compression of about one-third can be achieved.

PGP Operation - Segmentation/Reassembly

- E-mail facilities often are restricted to a maximum message length.
- To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail.
- The segmentation is done after all of the other processing, including the radix-64 conversion.
- Thus, the session key component and signature component appear only once, at the beginning of the first segment.
- Reassembly at the receiving end is required before verifying signature or decryption

S/MIME (Secure/Multipurpose Internet Mail Extension)

- S/MIME is a security enhancement to the MIME Internet e-mail format standard, which in turn provided support for varying content types and multi-part messages over the text only support in the original Internet RFC822 email standard.
- MIME allows encoding of binary data to textual form for transport over traditional RFC822 email systems.
- S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851 and S/MIME support is now included in many modern mail agents.

RFC 822

- RFC 822 defines a format for text messages that are sent using electronic mail and it has been the standard for Internet-based text mail message.

Multipurpose Internet Mail Extensions

- MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail

Problems with RFC 822 and SMTP

- Executable files or other binary objects must be converted into ASCII. Various schemes exist (e.g., Unix UUencode), but a standard is needed
- Text data that includes special characters (e.g., Hungarian text) cannot be transmitted as SMTP is limited to 7-bit ASCII
- Some servers reject mail messages over a certain size
- Some common problems exist with the SMTP implementations which do not adhere completely to the SMTP standards defined in RFC 821.
- They are:
- delete, add, or reorder CR and LF(Carriage Return/Line Feed) characters truncate or wrap lines longer than 76 characters remove trailing white space (tabs and spaces) pad lines in a message to the same length convert tab characters into multiple spaces

- MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations and the specification is provided in RFC's 2045 through 2049.

The MIME specification includes the following elements:

- Five new message header fields are defined, which provide information about the body of the message.
- A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
- Transfer encodings are defined that protect the content from alteration by the mail system.

UNIT V

IPSec – Internet Protocol Security

1. What is IPSec?

- IPSec = **Internet Protocol Security**.
- It's a set of standards that add **security to IPv4 & IPv6** using extra headers.
- Works at the **network (IP) layer** → so it protects *all applications* (email, web, file transfer, etc.) without changing them.
- Used for **VPNs (Virtual Private Networks)**, secure LAN, WAN, and Internet communication.

2. Main Goals of IPSec

IPSec provides **three core functions**:

1. **Authentication** – Verify the packet sender is genuine + ensure no tampering.
    - Uses **HMAC** (Hash-based Message Authentication Code).
    - Can protect:
        - Entire IP packet (**Tunnel Mode**)
        - Only the payload (**Transport Mode**)
2. **Confidentiality** – Keep data secret.
    - Uses encryption → **Encapsulating Security Payload (ESP)**.
    - Works in both **Tunnel** and **Transport** modes.
3. **Key Management** – Securely exchange, generate, and refresh encryption keys.

---

3. Applications of IPSec

- **VPNs** – Secure branch office connectivity across the Internet.
- **Remote access** – Employees/telecommuters connect securely from home/hotel.
- **Intranet/Extranet** – Secure communication with partners or subsidiaries.
- **E-commerce** – Adds stronger protection to online transactions.

---

4. Benefits of IPSec

- Protects **all traffic** going in/out of the network (if applied at firewall/router).
- Resistant to bypass (all Internet traffic must go through IPSec gateway).
- Transparent to applications → no software changes needed.
- Transparent to users → no training required.
- Supports both **organization-wide security** and **individual users**.
- Adds **routing security** (stops fake router updates, redirects, etc.).

5. IPSec Architecture

Defined by RFCs:

- **RFC 2401** – Security Architecture overview
- **RFC 2402** – Authentication Header (AH)
- **RFC 2406** – Encapsulating Security Payload (ESP)
- **RFC 2408** – Key Management (ISAKMP/IKE)

Protocols in IPSec

1. **Authentication Header (AH):**
   - Provides **authentication & integrity**.
   - Ensures packet not modified + verifies sender.
   - □ No encryption (data is visible).
2. **Encapsulating Security Payload (ESP):**
   - Provides **confidentiality (encryption)** + optional authentication.
   - Can encrypt payload (Transport Mode) or entire packet (Tunnel Mode).

Security Association (SA)

- **SA = One-way agreement** between sender & receiver about how to secure traffic.
- Defines which:
  - Protocol (AH or ESP)
  - Keys & algorithms
  - Key lifetime
- Each SA is identified by:
  1. **SPI (Security Parameters Index)**
  2. **Destination IP Address**
  3. **Protocol (AH or ESP)**
- Stored in **Security Association Database (SAD)**.
- For two-way secure communication → need **2 SAs**.

Security Policy Database (SPD)

- Decides which traffic needs IPSec protection.
- Uses **selectors** such as:
  - Source/Destination IP
  - Ports (TCP/UDP)
  - Protocol (TCP, UDP, ICMP)
  - User ID / Sensitivity level

Modes of Operation

1. **Transport Mode**
   - Protects **payload only** (leaves original IP header).
   - Used for **host-to-host** communication (e.g., client ↔ server).
2. **Tunnel Mode**
   - Protects the **entire original packet** by wrapping it in a new one.
   - Used for **gateway-to-gateway** (firewall ↔ firewall) or **host-to-gateway**.
   - Typical for **VPNs**.

IPSec Services Provided

✔ Access control
✔ Connectionless integrity
✔ Data origin authentication
✔ Replay protection (anti-replay window)
✔ Confidentiality (encryption)
✔ Limited traffic flow confidentiality

---

6. Example Scenario

LANs inside an organization run normal, unprotected IP.

- At the **boundary (firewall/router with IPSec)**:
  - Outgoing packets → encrypted & authenticated.
  - Incoming packets → decrypted & verified.
- Across the **public Internet**, packets remain secure.
- Remote users can also dial in securely if their systems support IPSec.

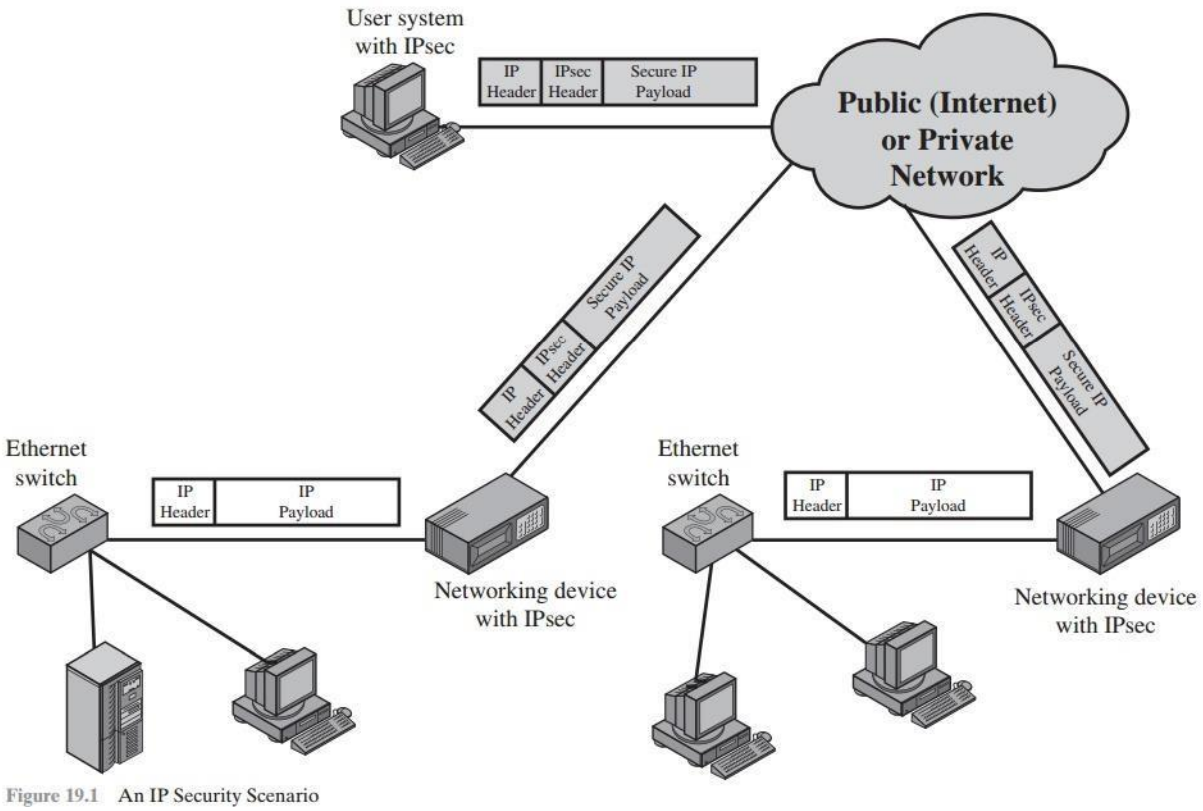


Figure 19.1 An IP Security Scenario

What the Diagram Shows

It explains how **IPsec (Internet Protocol Security)** protects communication between computers and network devices over the internet or any private network.

□□ Step-by-Step Explanation

1. **User System with IPsec**
   - This is a computer or user device that has **IPsec** enabled.
   - IPsec adds extra security information (like headers) to the data before it's sent.

   Data leaving the system looks like this:

   ```
   IP Header + IPsec Header + Secure IP Payload
   ```

- **IP Header** = Address label (shows source & destination)
- **IPsec Header** = Security lock (keeps data safe) AH & ESP
- **Secure IP Payload** = Actual message (locked/encrypted)

2. **Public or Private Network (like the Internet)**
   - This is the main communication path between devices.
   - Without IPsec, data travels as plain text (just IP Header + Payload).
   - With IPsec, the data is **encrypted and protected**.

   So even if someone intercepts it, they **can't read it**.

---

3. **Networking Devices with IPsec (Routers or Firewalls)**
   - These devices sit between networks and also have IPsec enabled.
   - They add IPsec protection to outgoing data and remove it for incoming data.

   Data between these devices travels securely:

   ```
   IP Header + IPsec Header + Secure IP Payload
   ```

---

4. **Ethernet Switches and Local Devices**
   - Inside a local network (like your office), data may not always be encrypted because it's within a trusted area.
   - But when data goes out to another network, IPsec kicks in to secure it.

   Inside LAN:

   ```
   IP Header + IP Payload (normal data)
   ```

   Over Internet:

```
IP Header + IPsec Header + Secure IP Payload (protected data)
```

A In Simple Words

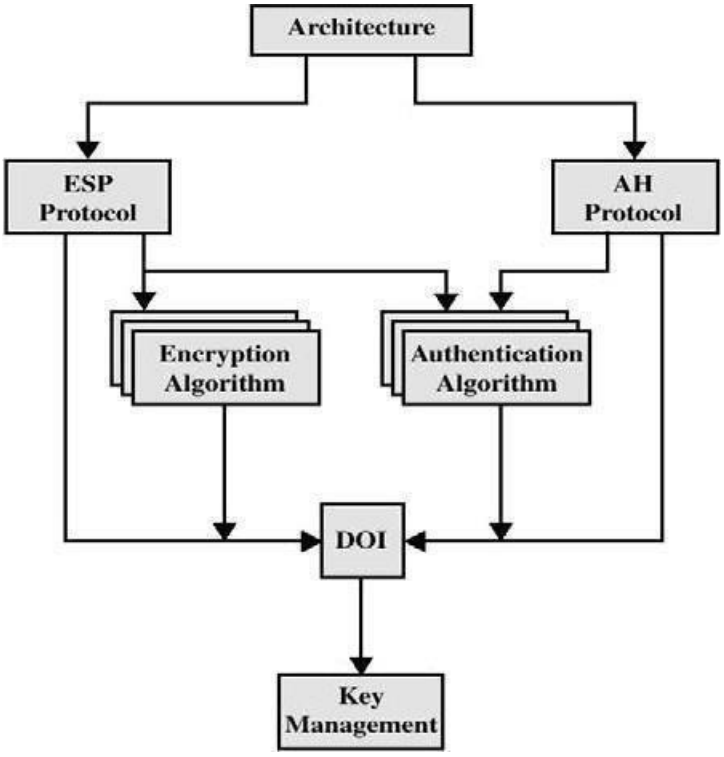
- **IPsec = Lock and Key for Internet Data**
- It makes sure your data is:
  - **Confidential** (nobody else can read it)
  - **Authentic** (comes from who it says it's from)
  - **Untampered** (not changed during transfer)

IPSec Overview

- **IPSec (Internet Protocol Security)** is a framework of protocols to secure communication over IP networks (like the Internet).
- It provides **Confidentiality, Integrity, Authentication, and Anti-replay protection**.
- Works at the **Network Layer (Layer 3)** → so it protects all applications automatically.

IPSec Architecture (as per your diagram)

1. **Architecture Layer**
   - The top-level design that connects all IPSec components.
2. **Two Core Protocols**
   - **AH (Authentication Header)** → Provides authentication & integrity (but no encryption).
   - **ESP (Encapsulating Security Payload)** → Provides encryption + authentication + integrity + anti-replay.
3. **Algorithms**
   - **Encryption Algorithms** → e.g., AES, 3DES (used by ESP for confidentiality).
   - **Authentication Algorithms** → e.g., HMAC-SHA, HMAC-MD5 (used by both AH & ESP for integrity and authentication).
4. **DOI (Domain of Interpretation)**
   - Defines rules & formats so all IPSec systems can understand each other.
   - Ensures interoperability between devices/vendors.
5. **Key Management**
   - Managed by **IKE (Internet Key Exchange)** protocol.
   - Handles creation, negotiation, and refreshing of keys for AH/ESP.



□ IPSec Modes of Operation

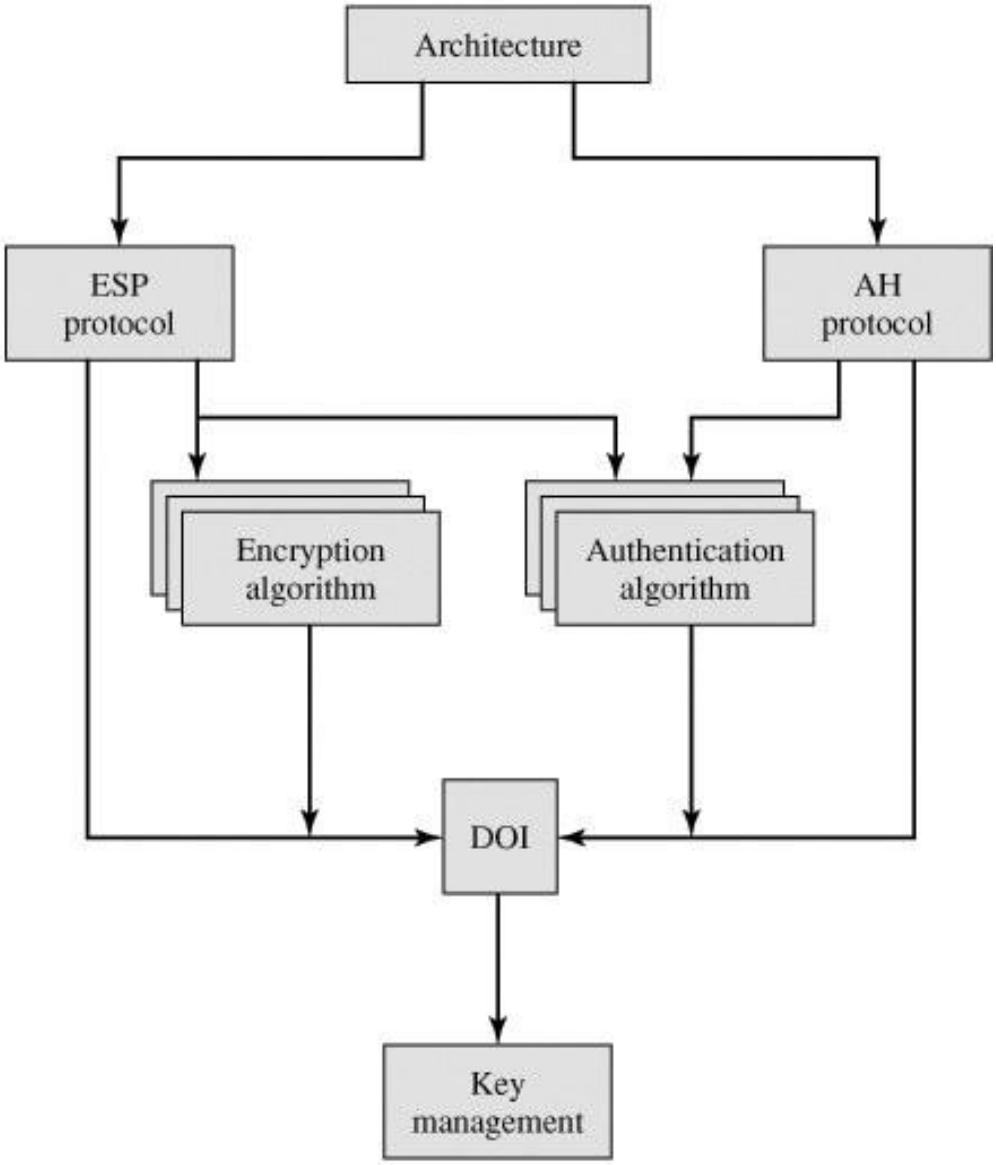
- **Transport Mode** → Protects only the payload (used end-to-end between hosts).
- **Tunnel Mode** → Protects the entire packet (used between gateways, e.g., VPNs).

□ IPSec Services

- **Confidentiality** → via encryption (ESP).
- **Integrity** → via hashing (AH, ESP).
- **Authentication** → ensures sender identity.
- **Anti-replay protection** → avoids packet duplication attacks.

□ In **simple words**:

- IPSec uses **AH & ESP protocols**.
- These protocols rely on **encryption + authentication algorithms**.
- **DOI** ensures everyone speaks the same security "language".
- **Key Management (IKE)** gives them secure keys.
- Together, they provide **secure tunnels over the Internet**.



Final Summary (Easy to Remember)

IPSec = AH + ESP + SA + SPD + Key Management

- **AH** → Authentication + Integrity (no encryption)
- **ESP** → Encryption + optional authentication
- **SA** → Agreement about how to secure traffic
- **SPD** → Decides which traffic gets IPSec
- **Modes** → Transport (payload) / Tunnel (whole packet)

IPSec = **Authentication + Confidentiality + Key Management at IP layer**, securing all apps and enabling VPNs.

Got it G — let's give a **clear explanation of your IPSec Architecture diagram** step by step, in very **easy terms**:

□ Explanation of IPSec Architecture Diagram

The diagram you uploaded shows how the **different parts of IPSec work together** to secure communication.

---

1. Architecture (Top Layer)

- This is the **overall design** of IPSec.
- It defines the framework that connects all the components (protocols, algorithms, key management).

---

2. Protocols

- IPSec has **two main protocols**:
  - **ESP (Encapsulating Security Payload):**
    → Provides **encryption** (hides data) + **authentication** (verifies sender).
  - **AH (Authentication Header):**
    → Provides **authentication + integrity** (ensures data not changed), but **no encryption**.

---

3. Algorithms

- These protocols depend on **mathematical algorithms**:
  - **Encryption Algorithms** (AES, DES, 3DES) → make data unreadable to outsiders.
  - **Authentication Algorithms** (HMAC-SHA, HMAC-MD5) → verify sender & detect changes.

So, ESP uses **encryption + authentication**, while AH uses only **authentication**.

---

4. DOI (Domain of Interpretation)

- Think of DOI as the **rulebook** that ensures all devices understand each other.
- Defines:
  - Which encryption algorithms are valid.
  - Which authentication methods are used.
  - How protocols should be implemented.

Without DOI, different vendors (Cisco, Juniper, Microsoft) might not be compatible.

---

5. Key Management

- Even the best encryption is useless without **secret keys**.
- IPSec uses **IKE (Internet Key Exchange)** to automatically:
  - Generate keys
  - Exchange them securely
  - Refresh them when needed
- This ensures **secure communication without manual setup**.

Simple Flow of the Diagram

1. **Architecture** defines the whole IPSec system.
2. **ESP & AH Protocols** provide security functions.
3. These protocols rely on **Encryption & Authentication Algorithms**.
4. **DOI** makes sure all IPSec systems understand each other.
5. Finally, **Key Management (IKE)** handles secret keys for encryption/authentication.

□ Encapsulating Security Payload (ESP) —

🔻 What is ESP?

* **ESP (Encapsulating Security Payload)** is a part of **IPSec (Internet Protocol Security)**.
* It **protects data** by providing:

1. **Confidentiality** → encrypts the data so others can't read it.
2. **Integrity** → ensures the data is not changed during transfer.
3. **Authentication** → confirms the data is from a trusted source.

□ Using encryption **without authentication** is not safe and is **discouraged**.

🔻 How ESP Works (Simple Explanation)

* ESP **encrypts only the data (payload)** part of the IP packet, **not the IP header**.
* It converts readable data into **unreadable (encrypted)** form to protect it from attackers.

* Only the **sender and authorized receiver** can decrypt and read the message.
* ESP uses **standard algorithms** like **SHA** and **MD5** for encryption and integrity checks.
* Each data packet has a **unique identifier (like a fingerprint)** so that fake or repeated packets can be detected.

🔷 Modes of ESP

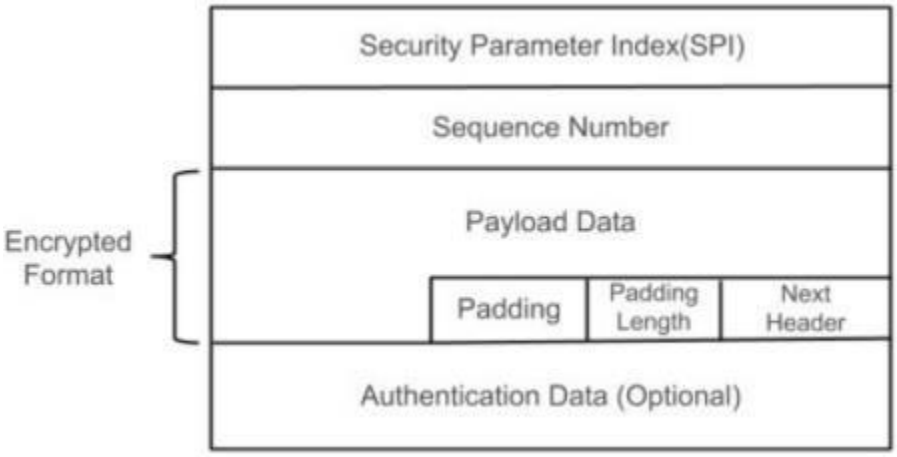
1. **Transport Mode:**

   * Encrypts only the **payload (data)**.
   * The **IP header is not encrypted.**
   * Used for **end-to-end communication** (e.g., between two computers).
   * ❓Does **not** provide integrity/authentication for the whole IP packet.

2. **Tunnel Mode:**

   * Encrypts the **entire IP packet (header + data)**.
   * Adds a **new IP header** outside the encrypted one.
   * Used between **gateways or VPNs**.

ESP Packet Format



Field Name Size Description (Easy Explanation)
Security Parameters Index (SPI) 32 bitsIdentifies the Security Association (SA) — like a unique connection ID used between sender and receiver.
Sequence Number 32 bitsA counter that increases with every packet; helps prevent replay attacks (duplicate packets).
Payload Data Variable The actual encrypted data — can be a transport-layer segment (in transport mode) or an IP packet (in tunnel mode).
Padding 0–255 bytes Extra bytes added to make the encrypted data fit the encryption block size properly.

Pad Length 8 bits Indicates how many padding bytes were added just before this field.
Next Header 8 bits Identifies the type of data in the payload (e.g., TCP, UDP, ICMP).
Authentication Data Variable Optional field used for integrity check — ensures the packet wasn't modified during transmission.

Authentication Header (AH) Packet Format

Purpose

- AH is used in IPsec to provide **data integrity**, **authentication**, and **anti-replay protection** for IP packets.
- It ensures that the **packet has not been modified** and that it **came from a valid source**.
- Unlike ESP, **AH does not provide encryption** (no confidentiality).

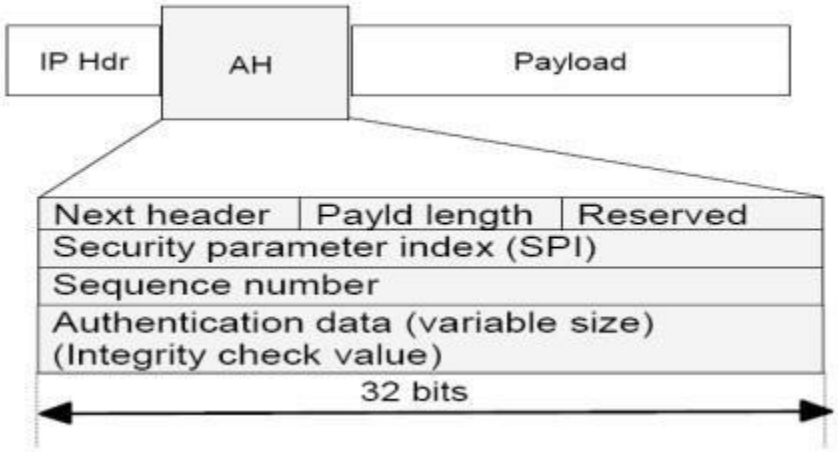
AH Packet Format (Field-wise Explanation)

1. **Next Header (8 bits)**
   - Identifies the **type of protocol** in the payload (e.g., TCP, UDP, ICMP).
   - Helps the receiver know which protocol header follows AH.
2. **Payload Length (8 bits)**
   - Indicates the **length of the AH header** in 32-bit words (excluding the first 2 fields).
   - Allows the receiver to determine where the payload starts.
3. **Reserved (16 bits)**
   - Set to **zero**.
   - Reserved for future use.
4. **Security Parameters Index (SPI) – 32 bits**
   - Uniquely identifies the **Security Association (SA)** between sender and receiver.
   - Tells which algorithms and keys to use for authentication.
5. **Sequence Number – 32 bits**
   - Incremented with each packet sent.
   - Used to **prevent replay attacks** by rejecting duplicate packets.
6. **Authentication Data (Variable length)**
   - Contains the **Integrity Check Value (ICV)** — a cryptographic hash (e.g., HMAC-SHA1).
   - Ensures that the **packet's header and payload** were not altered in transit.

Key Points to Remember

- AH protects both the **IP header** and the **payload** (unlike ESP, which mainly protects the payload).
- Provides:

    - **Integrity ✔**
    - **Authentication ✔**
    - **Anti-Replay Protection ✔**
    - **Confidentiality ✖** (No encryption)
- Works in **two modes:**
    1. **Transport Mode:** Protects the IP payload and parts of the IP header.
    2. **Tunnel Mode:** Protects the **entire original IP packet** inside a new IP header.
- Used when **data confidentiality is not required**, but **integrity and authenticity** are critical.



Intrusion detection system

Q What is an IDS?

- An IDS detects when someone tries to enter or misuse a computer system/network **without permission**.
- It watches network traffic or system behaviour, then alerts administrators if something seems wrong.
- Its job is **detection**, not prevention (it doesn't stop things by itself).

□ Common Methods of Intrusion

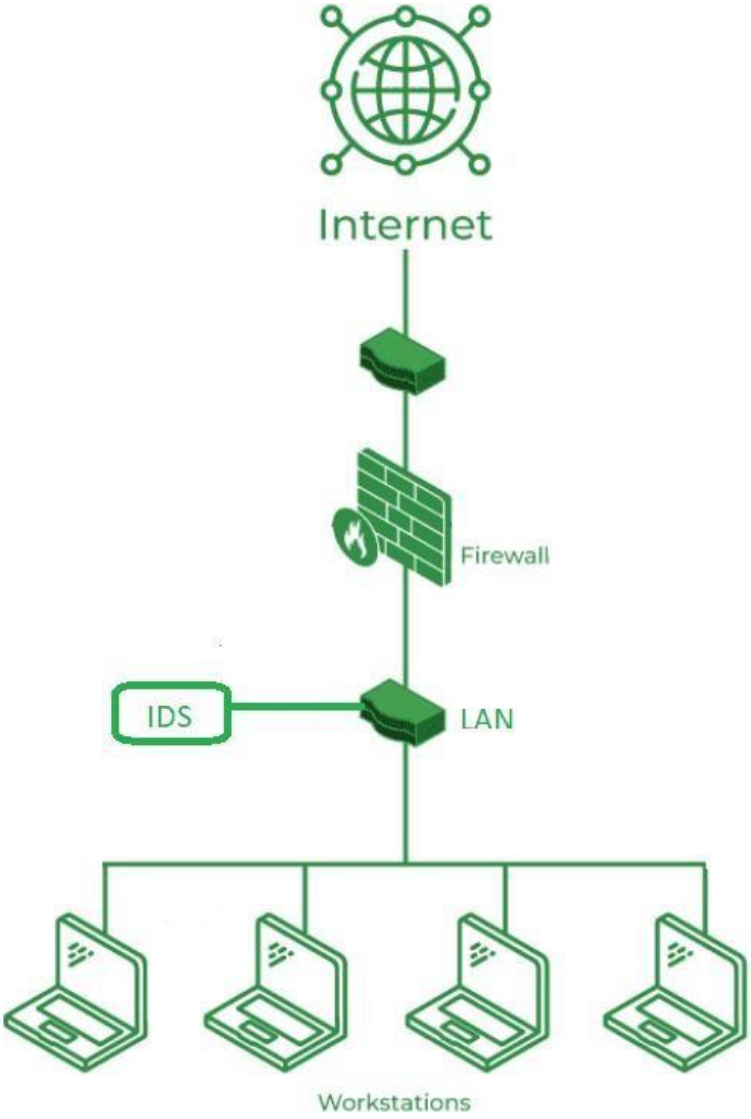
- **Address Spoofing**: Hiding the attacker's real address using a fake one.
- **Fragmentation**: Breaking data into small pieces so malicious content slips past detection.
- **Pattern Evasion**: Changing attack patterns so the IDS doesn't recognise them. (
- **Coordinated Attacks**: Multiple attacks/points used together to confuse detection.

□ Classification of IDS Types

1. **Network-based IDS (NIDS)**
   - Monitors network traffic from a central point. (
   - Example: Placed at a subnet to observe all traffic through a firewall. (
2. **Host-based IDS (HIDS)**
   - Runs on a single host (computer/server). Monitors that machine's incoming/outgoing packets + file changes. (
3. **Hybrid IDS**
   - Combines host- and network-based features for a more complete view.
4. **Application Protocol-based IDS (APIDS)**
   - Monitors application-specific protocols (e.g., SQL, HTTP) for misuse.
5. **Protocol-based IDS (PIDS)**
   - Sits at server front-end and interprets specific protocol streams (e.g., HTTP, HTTPS). (

□ How IDS Detects Intrusions

- Two major methods:
  - **Signature-based detection**: Looks for known patterns ("signatures") of attacks.
  - **Anomaly-based detection**: Builds a model of "normal" activity. Anything deviating from that model is flagged. (



□ Limitations / Challenges of IDS

- **False Alarms** – Many alerts may be benign, causing “alert fatigue”. (
- **Resource Intensive** – Requires CPU/memory + can slow down network if heavy. (
- **Does Not Prevent Attacks** – IDS only detects; stopping attacks needs other tools.
- **Complex to Manage** – Needs regular tuning, updates, expert oversight. (

□ Placement of IDS

- Commonly placed **behind the firewall**, so it can monitor traffic that passed the firewall and spot suspicious activity inside. (
- Can also be placed **inside the network** to watch internal traffic and detect internal threats.

SECURITY ASSOCIATIONS

Since IPSEC is designed to be able to use various secur ty protocols, it uses Security Associations (SA) to specify the protocols to be us . SA s a database record which specifies security parameters controlling security op rations. They are referenced by the sending host and established by the receiving host. An index parameter called the Security Parameters Index (SPI) is used. SAs are in one direction only and a second SA must be established for the transmis ion to be bi-directional. A security association is uniquely identified by three parameters:

• **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. The PI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

• **IP Destination Address**: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.

• **Security Protocol Identifier**: This indicates whether the association is an AH or ESP security association.

SA Parameters

In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:

Sequence Number Counter: A 32-bit value used to generate the Sequence Number field in AH or ESP headers

• **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations).

• **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is a replay

• **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations).

• **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).

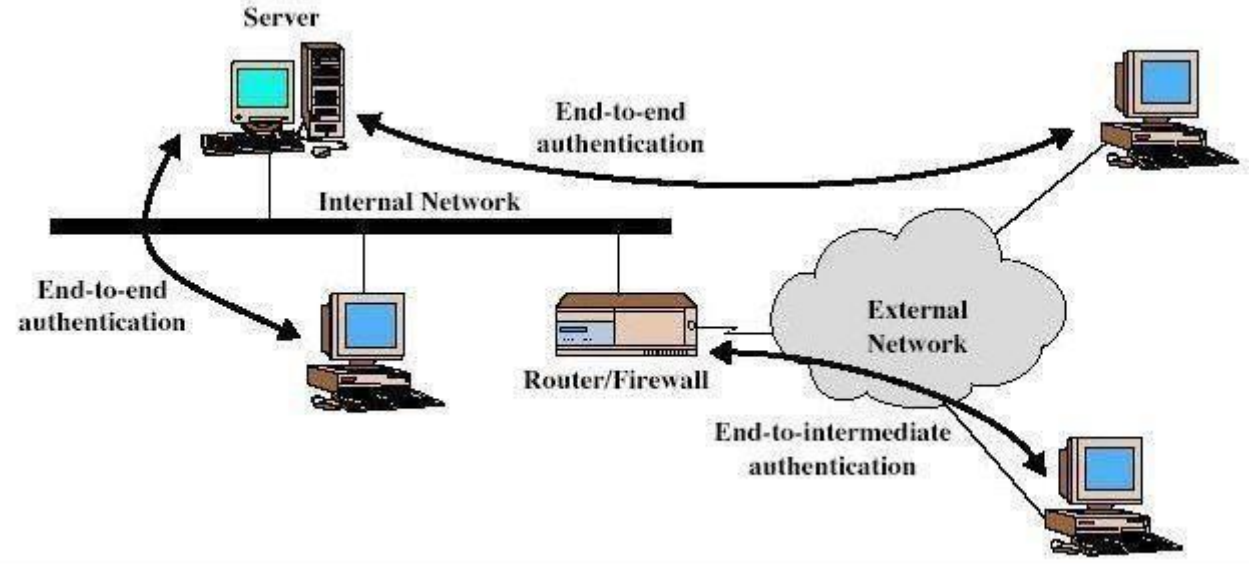
• **Lifetime of This Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indication of which of these actions should occur (required for all implementat ons).

• **IPSec Protocol Mode:** Tunnel, transport, or w ldcard (required for all implementations). These modes are discussed lat r in this section.

• **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).

Transport and Tunnel Modes

Both AH and ESP support two modes of use: transport and tunnel mode. **Transport Mode SA**		**Tunnel Mode SA**
AH	**Authenticates** IP payload and selected portions of IP header and IPv6 extension headers	**Authenticates** entire inner IP packet plus selected portions of outer IP header
ESP	**Encrypts** IP payload and any IPv6 extesion header	**Encrypts** inner IP packet
ESP with authentication	**Encrypts** IP payload and any IPv6 extesion header. **Authenticates** IP payload but no IP header	**Encrypts** inner IP packet. **Authenticates** inner IP packet



Key Management in Cryptography

Last Updated : 11 Jul, 2025

In cryptography, it is a very monotonous task to distribute the public and private keys between sender and receiver. If the key is known to the third party (forger/eavesdropper) then the whole security mechanism becomes worthless. So, there comes the need to secure the exchange of keys. In this article, we will learn about key management, how Cryptographic Keys Work**,** Types of Key Management, and Key Management Lifecycle.

What is Key Management?

Key management refers to the processes and procedures involved in generating, storing, distributing, and managing cryptographic keys used in cryptographic algorithms to protect sensitive data. It ensures that keys used to protect sensitive data are kept safe from unauthorized access or loss. Good key management helps maintain the security of encrypted information and is important for protecting digital assets from cyber threats. Effective key management is crucial for ensuring the confidentiality, integrity, and

availability of encrypted information by securing cryptographic keys from unauthorized access, loss, or compromise.

How Cryptographic Keys Works?

Cryptographic keys are special codes that protect information by locking (encrypting) and unlocking (decrypting) it. In **symmetric key cryptography**, a single shared key does both jobs, so the same key must be kept secret between users. In **asymmetric key cryptography**, there are two keys: a public key that anyone can use to encrypt messages or verify signatures, and a private key that only the owner uses to decrypt messages or create signatures. This makes it easier to share the public key openly while keeping the private key secret. These keys are crucial for secure communication, like when you visit a secure website (HTTPS), where they help encrypt your data and keep it safe from eavesdroppers and criminals. So, to manage these keys properly is vital to keep digital information secure and dependable.

Types of Key Management

There are two aspects of Key Management:

1. Distribution of public keys.
2. Use of public-key encryption to distribute secrets.

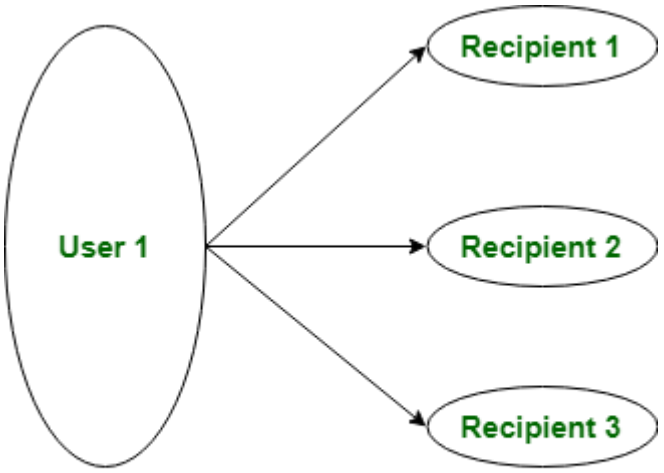
Distribution of Public Key

The public key can be distributed in four ways:

1. Public announcement
2. Publicly available directory
3. Public-key authority
4. Public-key certificates.

These are explained as following below:

1. Public Announcement: Here the public key is broadcast to everyone. The major weakness of this method is a forgery. Anyone can create a key claiming to be someone else and broadcast it. Until forgery is discovered can masquerade as claimed user.



Public Key Announcement

2. Publicly Available Directory: In this type, the public key is stored in a public directory. Directories are trusted here, with properties like Participant Registration, access and allow to modify values at any time, contains entries

like {name, public-key}. Directories can be accessed electronically still vulnerable to forgery or tampering.

3. Public Key Authority: It is similar to the directory but, improves security by tightening control over the distribution of keys from the directory. It requires users to know the public key for the directory. Whenever the keys are needed, real-time access to the directory is made by the user to obtain any desired public key securely.

4. Public Certification: This time authority provides a certificate (which binds an identity to the public key) to allow key exchange without real-time access to the public authority each time. The certificate is accompanied by some other info such as period of validity, rights of use, etc. All of this content is signed by the private key of the certificate authority and it can be verified by anyone possessing the authority's public key.
First sender and receiver both request CA for a certificate which contains a public key and other information and then they can exchange these certificates and can start communication.

Key Management Lifecycle

The **key management lifecycle** outlines the stages through which cryptographic keys are generated, used, and eventually retired or destroyed. Proper management of these keys is critical to ensuring the security of cryptographic systems. Here's an overview of each stage:

1. Key Generation:

- **Creation**: Keys are created using secure algorithms to ensure randomness and strength.
- **Initialization**: Keys are initialized with specific parameters required for their intended use (e.g., length, algorithm).

2. Key Distribution:

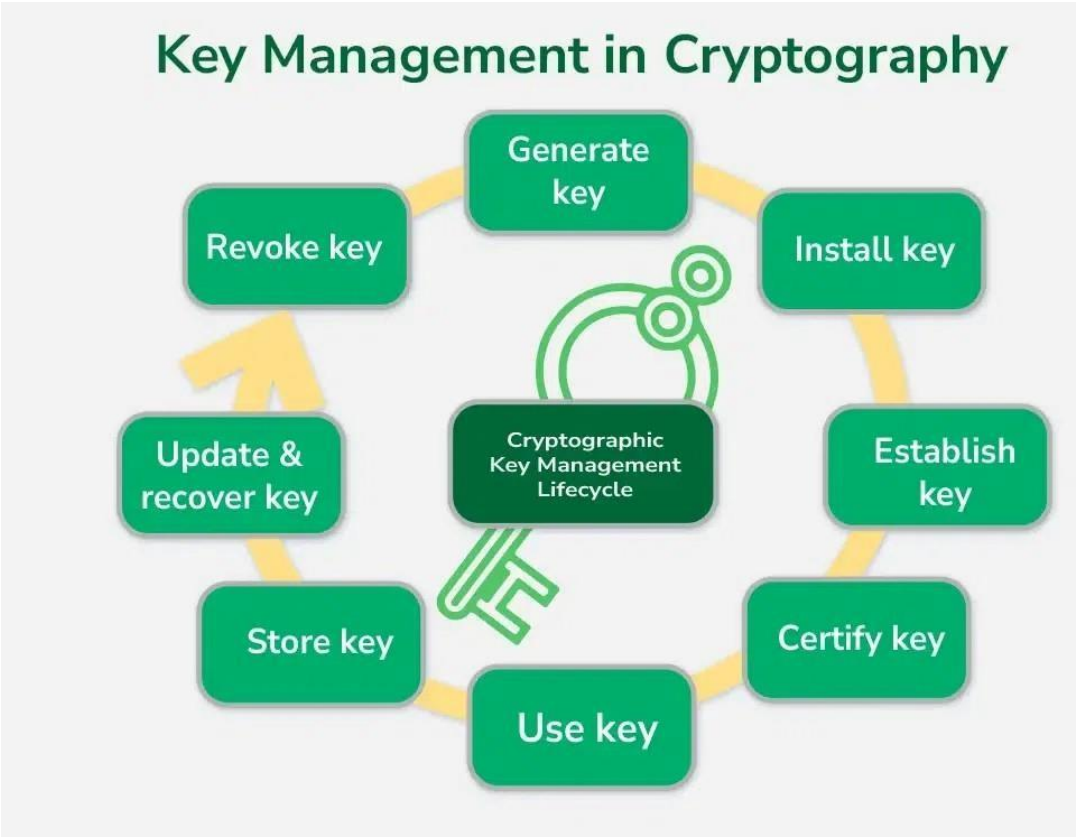
- **Sharing**: For symmetric keys, secure methods must be used to share the key between parties.
- **Publication**: For asymmetric keys, the public key is shared openly, while the private key remains confidential.

3. Key Storage:

- **Protection**: Keys must be stored securely, typically in hardware security modules (HSMs) or encrypted key stores, to prevent unauthorized access.
- **Access Control**: Only authorized users or systems should be able to access keys.

4. Key Usage:

- **Application**: Keys are used for their intended cryptographic functions, such as encrypting/decrypting data or signing/verifying messages.
- **Monitoring**: Usage is monitored to detect any unusual or unauthorized activities.



Key Management in Cryptography

5. Key Rotation:

- **Updating**: Keys are periodically updated to reduce the risk of exposure or compromise.
- **Re-Keying**: New keys are generated and distributed, replacing old ones while ensuring continuity of service.

6. Key Revocation:

- **Invalidation**: Keys that are no longer secure or needed are invalidated.
- **Revocation Notices**: For public keys, revocation certificates or notices are distributed to inform others that the key should no longer be trusted.

7. Key Archival:

- **Storage**: Old keys are securely archived for future reference or compliance purposes.
- **Access Restrictions**: Archived keys are kept in a secure location with restricted access.

8. Key Destruction:

- **Erasure**: When keys are no longer needed, they are securely destroyed to prevent any possibility of recovery.
- **Verification**: The destruction process is verified to ensure that no copies remain.

Conclusion

Managing cryptographic keys is crucial for keeping data secure. It involves creating, distributing, storing, using, updating, and eventually destroying keys properly. Good key management ensures that keys are safe from

unauthorized access and can be trusted throughout their life. By doing this, organizations protect sensitive information and maintain the security of their digital communications. In short, effective key management is essential for making encryption work and keeping information systems secure.