UNIT – I

1. Computer and Network Security Concepts:

1.1. Computer Security Concepts

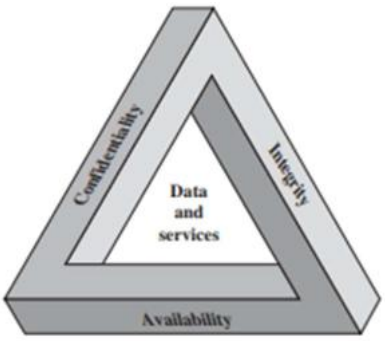
A Definition of Computer Security: The NIST Computer Security Handbook [NIST95] defines the term computer security as follows:

"The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications)".

This definition introduces three key objectives that are at the heart of computer security:

- Confidentiality: This term covers two related concepts:
  - Data confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.
  - Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
- Integrity: This term covers two related concepts:
  - Data integrity: Assures that information and programs are changed only in a specified and authorized manner.
  - System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.
- Availability: Assures that systems work promptly and service is not denied to authorized users

These three concepts form what is often referred to as the CIA triad



The three concepts embody the fundamental security objectives for both data and for information and computing services.

• **Confidentiality**: Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.

• **Integrity**: Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.

• **Availability**: Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.

Although the use of the CIA triad to define security objectives is well established, some in the security field feel that additional concepts are needed to present a complete picture. Two of the most commonly mentioned are as follows:

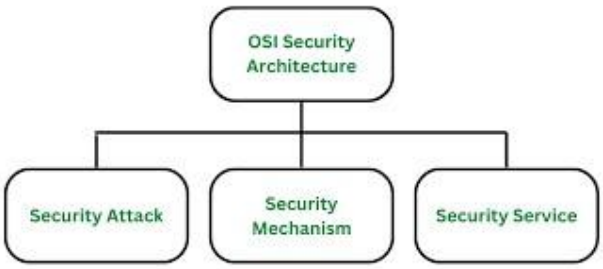
- **Authenticity**: The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the system came from a trusted source.
- **Accountability**: The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action. Because truly secure systems are not yet an achievable goal, we must be able to trace a security breach to a responsible party. Systems must keep records of their activities to permit later forensic analysis to trace security breaches or to aid in transaction disputes.

1.2. The OSI Security Architecture

ITU-T3 Recommendation X.800, Security Architecture for OSI, defines such a systematic approach.4 The OSI security architecture is useful to managers as a way of organizing the task of providing security.

The Open System Interconnection (OSI) security architecture is a recommendation of the International Telecommunication Union which defines a systematic approach to define security requirements for a certain organization, as well as approaches to meet the aforementioned requirements.

The OSI security architecture provides a general description of security services and mechanisms, as well as a description of security attacks.



Security attack: Any action that compromises the security of information owned by an organization.

Security mechanism: A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

Security service: A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

Threat: A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Attack: An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

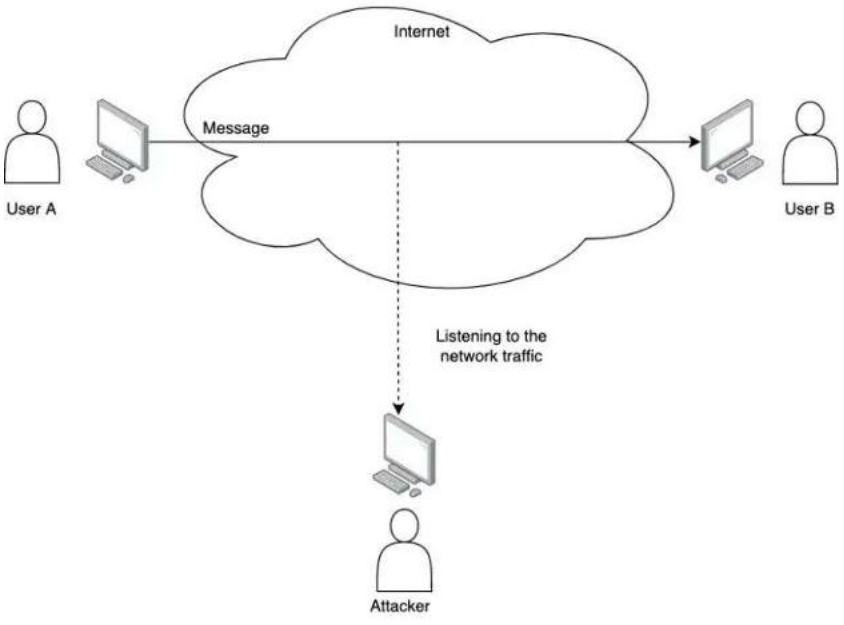
In the literature, the terms threat and attack are commonly used to mean more or less the same thing.

1.3. Security Attacks

A security attack is an activity or act made upon a system with the goal to obtain unauthorized access to information or resources. It is usually carried out by evading security policies that are in place in organizations or individual devices.

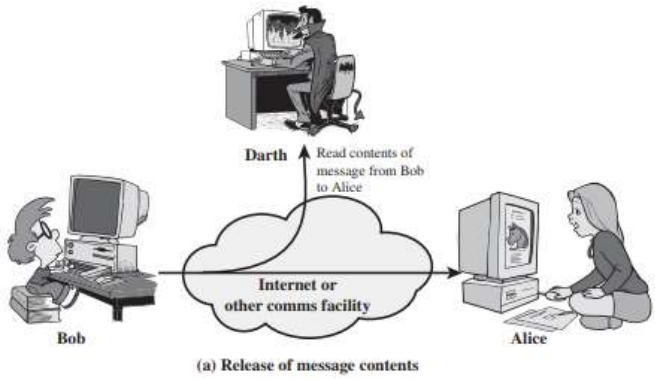
Security attacks can be passive or active.

A passive security attack is one where the attacker just listens to or monitors a communication. In this case, the attacker will not impersonate any of the parts of the communication. See an example in the figure below.



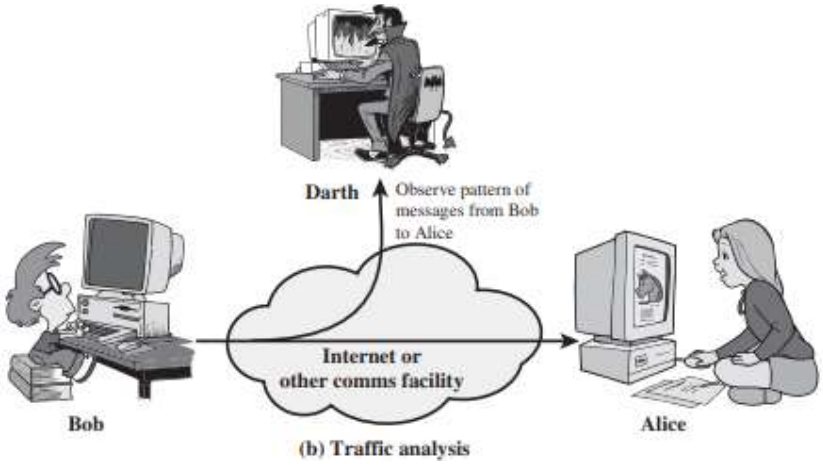
Some of the passive attacks are:

Release of message contents: This happens when the attacker can intercept confidential information: bank statements/account documents, payslips, etc.



(a) Release of message contents

Traffic analysis: In this case, the attacker just listens to the traffic to gain knowledge that is not public. As an example, an attacker is listening to a communication between a famous actress and her manager.

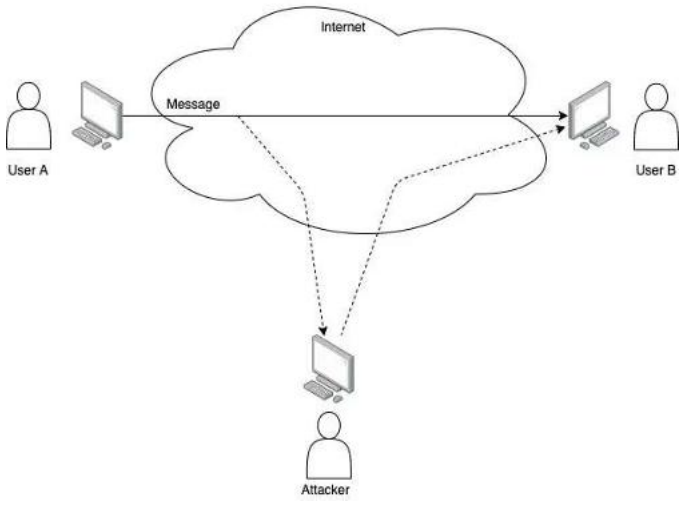


(b) Traffic analysis

The way to prevent this type of attack is by using encryption. If the communication is encrypted using secure mechanisms, then it does not matter if an attacker listens to the communication, because the attacker won't be able to make any sense of it.

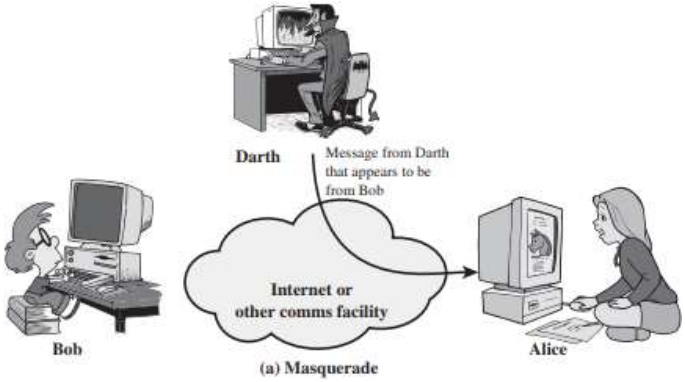
On the other hand, is an active attack, the attacker will modify the information that is transmitted and try to impersonate one or both parties in the communication. Another type of attack is when the attacker creates a message and starts a communication pretending to be someone else in order to gain access to sensitive information, get someone to do something, or cause a denial of service.

See a general representation below.

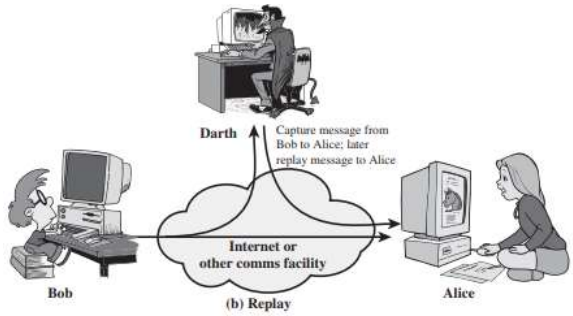


Some examples of active attacks are:

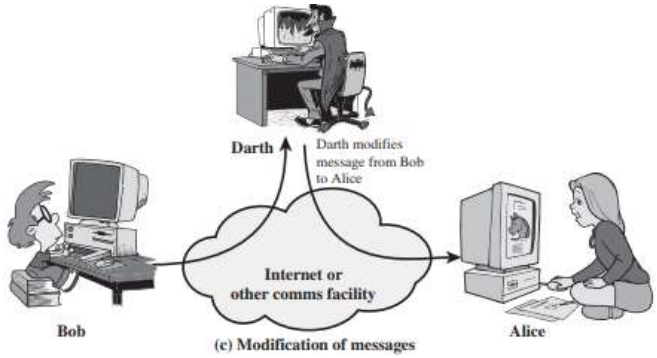
Masquerade attack. When the attacker impersonates someone else. This type of attack is usually used together with another attack, like reply and modification attacks.



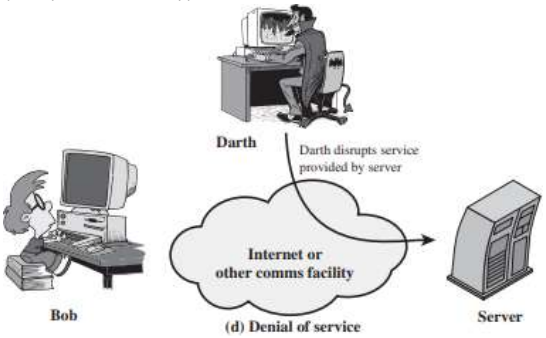
Reply attack. In this type of attack the attacker capture the data that is transmitted and re-sends it to produce an unauthorized action.



Modification of messages. This type of attack happens when the attacker capture information, modifies it, and re-sends it to produce an unauthorized action.



Denial of service. This type of attack occurs when a service is prevented to work as expected. Usually, it also brings as consequence, other services or users are also prevented to do their duties. As an example, imagine that an e-mail is sent to a person with instructions to perform a time-sensitive task. At the same time, a Denial of Service (DoS) attack is performed on the email server. As a consequence, the person cannot read the email (the server is down) and cannot carry on the instructions (the person's duty).



1.4. Security Services

X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers.

a security service is “A processing or communication service that is provided by a system to give a specific kind of protection to system resources”.

X.800 divides these services into five categories and fourteen specific services

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication
Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data-Origin Authentication
In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality
The protection of all user data on a connection.

Connectionless Confidentiality
The protection of all user data in a single data block

Selective-Field Confidentiality
The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic-Flow Confidentiality
The protection of the information that might be derived from observation of traffic flows.

DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery
Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery
As above, but provides only detection without recovery.

Selective-Field Connection Integrity
Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity
Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity
Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

NONREPUDIATION

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

Nonrepudiation, Origin
Proof that the message was sent by the specified party.

Nonrepudiation, Destination
Proof that the message was received by the specified party.

1.5. Security Mechanisms

A security mechanism is "A method or process (or a device incorporating it) that can be used in a system to implement a security service that is provided by or within the system".

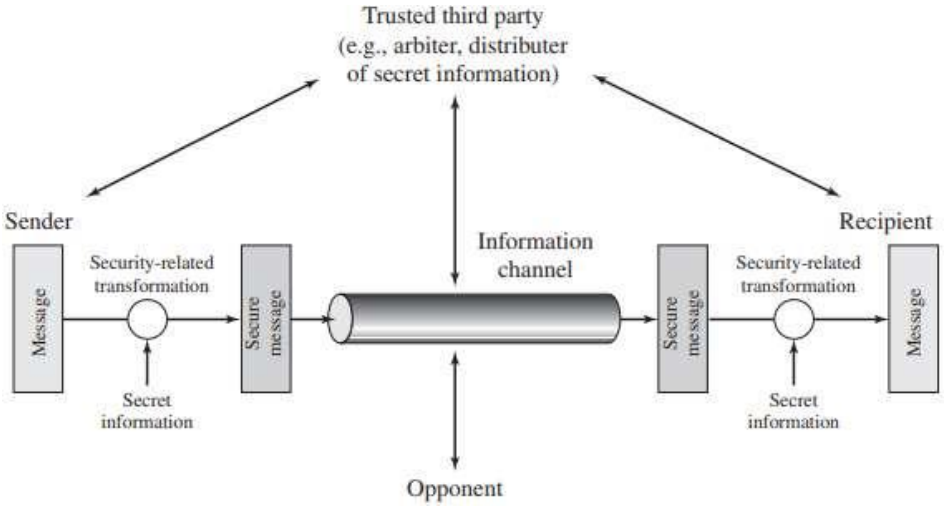
Some examples are authentication exchange, checksum, digital signature, encryption, and traffic padding.

Table below lists the security mechanisms defined in X.800. The mechanisms are divided into those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol, and those that are not specific to any particular protocol layer or security service.

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.	Mechanisms that are not specific to any particular OSI security service or protocol layer.
Encipherment The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.	**Trusted Functionality** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
Digital Signature Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).	**Security Label** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
Access Control A variety of mechanisms that enforce access rights to resources.	**Event Detection** Detection of security-relevant events.
Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.	**Security Audit Trail** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.	**Security Recovery** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.
Traffic Padding The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.	
Routing Control Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.	
Notarization The use of a trusted third party to assure certain properties of a data exchange.	

1.6. A Model for Network Security

Figure below shows Model for Network Security:



A message is to be transferred from one party to another across some sort of Internet service.

The two parties, who are the principals in this transaction, must cooperate for the exchange to take place.

A logical information channel is established by defining a route through the Internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

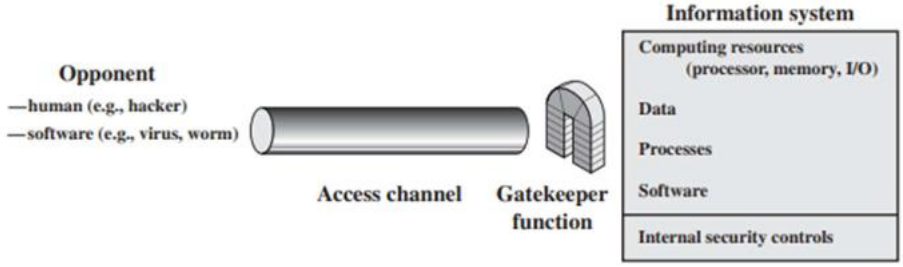
- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

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

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
2. Generate the secret information to be used with the algorithm.
3. Develop methods for the distribution and sharing of the secret information.
4. 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.

A general model of these other situations is illustrated by Figure below (Network Access Security Model or System Security), which reflects a concern for protecting an information system from unwanted access.



Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers. Programs can present two kinds of threats:

• Information access threats: Intercept or modify data on behalf of users who should not have access to that data.

• Service threats: Exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security

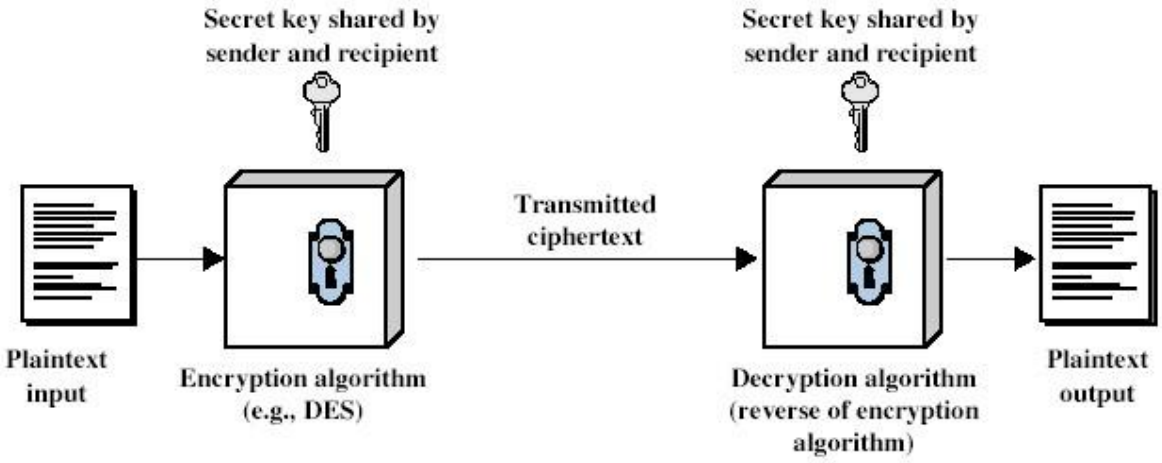
2. Classical Encryption Techniques:

Classical encryption techniques refer to traditional methods of encrypting information that were used before the advent of modern cryptographic algorithms and computer-based encryption systems. Classical Encryption Techniques are the basic building blocks of all encryption techniques.

The two basic building blocks of all encryption techniques are substitution and transposition.

2.1. Symmetric Cipher Model

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of publickey encryption in the 1970s.



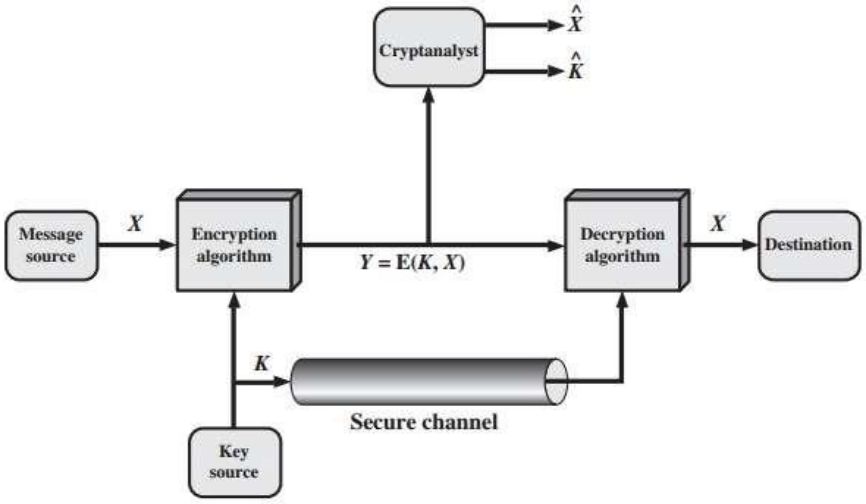
A symmetric encryption scheme has five ingredients

1. **Plain Text (x)**: This is the original data/message that is to be communicated to the receiver by the sender. It is one of the inputs to the encryption algorithm.
2. **Secret Key (k)**: It is a value/string/text file used by the encryption and decryption algorithm to encode and decode the plain text to cipher text and vice-versa respectively. It is independent of the encryption algorithm. It governs all the conversions in plain text. All the substitutions and transformations done depend on the secret key.
3. **Encryption Algorithm (E)**: It takes the plain text and the secret key as inputs and produces Cipher Text as output. It implies several techniques such as substitutions and transformations on the plain text using the secret key. E (x, k) = y
4. **Cipher Text (y)**: It is the formatted form of the plain text (x) which is unreadable for humans, hence providing encryption during the transmission. It is completely dependent upon the secret key provided to the encryption algorithm. Each unique secret key produces a unique cipher text.
5. **Decryption Algorithm (D)**: It performs reversal of the encryption algorithm at the recipient's side. It also takes the secret key as input and decodes the cipher text received from the sender based on the secret key. It produces plain text as output. D (y, k) = x.

There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm.
2. The sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure below:



A source produces a message in plaintext, X = [X1, X2, ... , XM]. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet {0, 1} is typically used. For encryption, a key of the form K = [K1, K2, ... , KJ] is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext Y = [Y1, Y2, ... , YN]. We can write this as Y = E(K, X).

This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X , with the specific function determined by the value of the key K .

The intended receiver, in possession of the key, is able to invert the transformation: X = D(K, Y)

An opponent, observing Y but not having access to K or X , may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption.

2.2. Substitution Techniques

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

2.2.1. Caesar Cipher

The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.

For example,

meet	me	after	the	toga	party
PHHW	PH	DIWHU	WKH	WRJD	SDUWB

We can define the transformation by listing all possibilities, as follows:

plain:	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
cipher:	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Let us assign a numerical equivalent to each letter:

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
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter, substitute the ciphertext letter:

C = E (3, p) = (p + 3) mod 26

A shift may be of any amount, so that the general Caesar algorithm is

$C = E(k, p) = (p + k) \bmod 26$

where takes on a value in the range 1 to 25. The decryption algorithm is simply

$p = D(k, C) = (C - k) \bmod 26$

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: simply try all the 25 possible keys.

Figure below shows the results of applying this strategy to the example ciphertext.

KEY	PHHW PH DIWHU WKH WRJD SDUWB
1	oggv og chvgt vjg vqic rctva
2	nffu nf bgufs uif uphb qbsuz
3	meet me after the toga party
4	ldds ld zesdq sgd snfz ozqsx
5	kccr kc ydrcp rfc rmey nyprw
6	jbbq jb xcqbo qeb qldx mxoqv
7	iaap ia wbpan pda pkcw lwnpu
8	hzzo hz vaozm ocz ojbv kvmot
9	gyyn gy uznyl nby niau julns
10	fxxm fx tymxk max mhzt itkmr
11	ewwl ew sxlwj lzw lgys hsjlq
12	dvvk dv rwkvi kyv kfxr grikp
13	cuuj cu qvjuh jxu jewq fqhjo
14	btti bt puitg iwt idvp epgin
15	assh as othsf hvs hcuo dofhm
16	zrrg zr nsgre gur gbtn cnegl
17	yqqf yq mrfqd ftq fasm bmdfk
18	xppe xp lqepc esp ezrl alcej
19	wood wo kpdob dro dyqk zkbdi
20	vnnc vn jocna cqn cxpj yjach
21	ummb um inbmz bpm bwoi xizbg
22	tlla tl hmaly aol avnh whyaf
23	skkz sk glzkx znk zumg vgxze
24	rjjy rj fkyjw ymj ytlf ufwyd
25	qiix qi ejxiv xli xske tevxc

2.2.2.Monoalphabetic Ciphers

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution.

A permutation of a finite set of elements is an ordered sequence of all the elements S of, with each element appearing exactly once. For example, if S={a, b, c}, there are six permutations of :

abc, acb, bac, bca, cab, cba

In general, there are n! permutations of a set of elements, because the first element can be chosen in one of n ways, the second in ways, the third in ways, and so on.

- **Substitution Cipher**: A monoalphabetic cipher is a type of substitution cipher, where each letter in the plaintext (original message) is replaced with a different letter in the ciphertext (encrypted message).
- **Single Alphabet**: It uses a single, fixed mapping for the entire encryption process. This means each letter in the plaintext always maps to the same letter in the ciphertext.

Key Creation:

Key: The key is the mapping between the plaintext alphabet and the ciphertext alphabet. It determines how each letter is substituted.

Example Key: Here's an example key:

Plaintext Alphabet : ABCDEFGHIJKLMNOPQRSTUVWXYZ

Ciphertext Alphabet : QWERTZUIOPASDFGHJKLYXCVBNM

Encryption Process:

Replace Letters: To encrypt a message, you simply replace each letter in the plaintext with its corresponding letter in the ciphertext, according to the key.

Example Encryption: Plaintext: MEET ME AT THE PARK Ciphertext: ZGGZ ZG ZG KJC YGKZ

Decryption Process:

Reverse Substitution: To decrypt the ciphertext, you reverse the substitution process using the same key.

Example Decryption: Ciphertext: ZGGZ ZG ZG KJC YGKZ Plaintext: MEET ME AT THE PARK

Security Considerations:

Frequency Analysis: Monoalphabetic ciphers are relatively easy to break using frequency analysis, which involves analyzing the frequency of letters in the ciphertext to try to determine the plaintext letters.

2.2.3. Polyalphabetic cipher

A polyalphabetic cipher is a more advanced form of encryption where each letter in the plaintext can be mapped to multiple ciphertext letters, depending on its position in the message. The most famous polyalphabetic cipher is the Vigenère cipher.
Example:
Suppose we want to encrypt the plaintext message "HELLO" using a Vigenère cipher with the key "KEY."
Step 1: Key Repetition
Repeat the key so that it matches the length of the plaintext message.

Plaintext: H E L L O
Key: K E Y K E
Key Stream: K E Y K E

Step 2: Mapping
Create a table that shows the mapping between each letter in the plaintext alphabet and its corresponding letter in the ciphertext alphabet for each position in the key stream.

Plaintext: 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
Key Stream 1: K E Y K E Y K E Y K E Y K E Y K E Y K E Y K E Y
Ciphertext 1: K F A L P J O R W C U S X H M B D V N I T Q G Z

Plaintext: 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
Key Stream 2: K E Y K E Y K E Y K E Y K E Y K E Y K E Y K E Y
Ciphertext 2: O X M A G Z V L H S P C N Q W J D Y R K E U I F B T

Plaintext: 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
Key Stream 3: K E Y K E Y K E Y K E Y K E Y K E Y K E Y K E Y
Ciphertext 3: R W C U S X H M B D V N I T Q G Z K F A L P J O

Plaintext: 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
Key Stream 4: K E Y K E Y K E Y K E Y K E Y K E Y K E Y K E Y
Ciphertext 4: O X M A G Z V L H S P C N Q W J D Y R K E U I F B T

Step 3: Encryption

Encrypt each letter in the plaintext by finding the intersection of its row and the column corresponding to the letter in the key stream.

Plaintext: H E L L O
Key Stream: K E Y K E
Ciphertext: K X C U G

So, the encrypted message for "HELLO" using the Vigenère cipher with the key "KEY" is "KXCUF."

Step 4: Decryption
To decrypt a message encrypted with the Vigenère cipher, you would reverse the process using the same key. You would use the key stream to find the intersection of the ciphertext and determine the original plaintext.

Polyalphabetic ciphers are more secure than monoalphabetic ciphers, but they are not immune to attacks.

2.2.4. Playfair Cipher

The Playfair cipher is a type of polyalphabetic substitution cipher that encrypts pairs of letters instead of single letters. Here's how to encrypt the plaintext "HELLO" using the Playfair cipher:

1. Choose a keyword. For example, we can use the keyword MONARCHY.
2. Write down the plaintext message. In this case, it is HELLO.
3. Split the plaintext into pairs of two letters (digraphs). If there is an odd number of letters, add a Z to the last letter. In this case, the plaintext becomes HE LZ LO.
4. Generate the key square (5x5 grid) using the keyword. The key square is a 5x5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I. The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order. Here's the key square generated using the keyword MONARCHY:

	1	2	3	4	5
1	M	O	N	A	R
2	C	H	Y	B	D
3	E	F	G	I	K
4	L	P	Q	S	T
5	U	V	W	X	Z

5. Encrypt each digraph using the key square. If the two letters of the digraph are in the same row, replace each letter with the letter to its right (wrapping around to the leftmost letter if necessary). If the two letters of the digraph are in the same column, replace each letter with the letter below it (wrapping around to the topmost letter if necessary). If the two letters of the digraph are neither in the same row nor in the same column, replace each letter with the letter in the same row but in the column of the other letter of the digraph. For example, the digraph HE is encrypted as DL, the digraph LZ is encrypted as TP, and the digraph LO is encrypted as OK. The resulting ciphertext is DLTPOK.

To decrypt the ciphertext, simply reverse the process by replacing each digraph with the corresponding plaintext digraph. For example, DL becomes HE, TP becomes LZ, and OK becomes LO. The resulting plaintext is HELLO.

2.3. Transposition Techniques

A transposition cipher is a type of encryption method that maps plain text into cipher text by performing permutation over the plain text.

2.3.1. Rail Fence Cipher

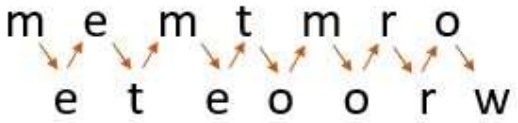
The rail fence cipher is the simplest transposition cipher. The steps to obtain cipher text using this technique are as follow:

Step 1: The plain text is written as a sequence of diagonals.

Step 2: Then, to obtain the cipher text the text is read as a sequence of rows.

Plain Text: meet me Tomorrow

Now, we will write this plain text sequence wise in a diagonal form as you can see below:



Once you have written the message as a sequence of diagonals, to obtain the cipher text out of it you have to read it as a sequence of rows. So, reading the first row the first half of cipher text will be:

m e m t m r o

reading the second row of the rail fence, we will get the second half of the cipher text:

e t e o o r w

Now, to obtain the complete cipher text combine both the halves of cipher text and the complete cipher text will be:

Cipher Text: M E M T M R O E T E O O R W

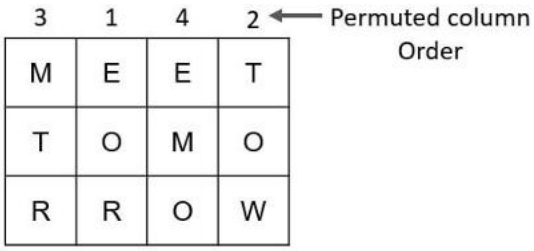
2.3.2. Columnar Transposition Technique

The columnar transposition cipher is more complex as compared to the rail fence. The steps to obtain cipher text using this technique are as follow:

Step 1: The plain text is written in the rectangular matrix of the initially defined size in a row by row pattern.
Step 2: To obtain the cipher text read the text written in a rectangular matrix column by column. But you have to permute the order of column before reading it column by column. The obtained message is the cipher text message.

To understand the columnar transposition let us take an example:
Plain text: meet Tomorrow

Now, put the plain text in the rectangle of a predefined size. For our example, the predefined size of the rectangle would be 3×4. As you can see in the image below the plain text is placed in the rectangle of 3×4. And we have also permuted the order of the column.



Now, to obtain the cipher text we have to read the plain text column by column as the sequence of permuted column order. So, the cipher text obtained by the columnar transposition technique in this example is:

Cipher Text: MTREOREMOTOW.

Similar to the rail fence cipher, the columnar cipher can be easily broken. The cryptanalyst only has to try few permutation and combination over the order of column to obtain the permuted order of column and the get the original message.

2.3.3. Book Cipher or Running Key Cipher

The book cipher or the running key cipher works on the basic principle of one-time pad cipher. In onetime pad cipher the key is taken as long as the plain text and is discarded after the use. Every time a new key is taken for a new message.

The improvement to the onetime pad in Book cipher is that the key or the onetime pad is taken from the book. Let us discuss the steps:
Step 1: Convert the plain text in numeric form consider A=0, B=1, C=3 …, Z=25.

Step 2: Take an onetime pad or key from any of the books and convert it in the numeric form also. But the key must be as long as the length of plain text.

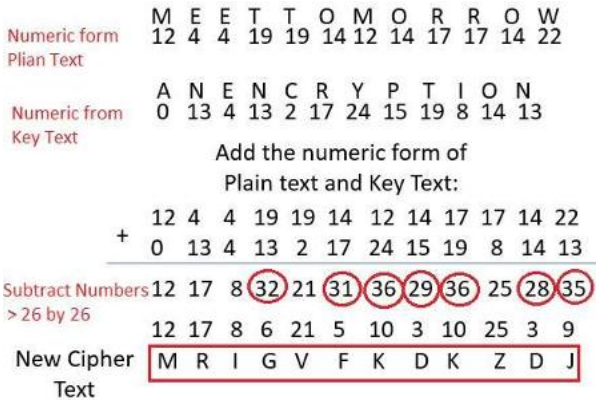
Step 3: Now add the numeric form of both plain text and key, each plain text letter with corresponding key text letter. If the addition of any plain text letter with corresponding key text letter is >26, then subtract it with 26.

Let us understand with the example:

Plain text: Meet Tomorrow

Key taken from the book: ANENCRYPTION.

Now we have to convert this plain text and key text in numeric form and add them to get cipher text as shown in the image below:



The cipher text obtained is MRIGVFKDKZDJ.

2.4. Steganography

Steganography is the practice of concealing a message within another message or a file. It is a technique used to hide secret information in plain sight. There are several types of steganography techniques, including:

1. Text Steganography: In this technique, the hidden data is encoded into the letter of each word. For example, the first letter of each word in a sentence can be used to hide a message.

2. Image Steganography: In this technique, a message is embedded into an image by altering the values of some pixels, which are chosen by an encryption algorithm. The recipient of the image must be aware of the same algorithm in order to know which pixels he or she must select to extract the message. For example, a picture of a tree can be used to hide another image.

3. Audio Steganography: In this technique, a message is hidden in an audio file by altering the sound waves. The hidden message can be revealed by analysing the sound waves.

4. Video Steganography: In this technique, a message is hidden in a video file by altering the frames. The hidden message can be revealed by analysing the frames.

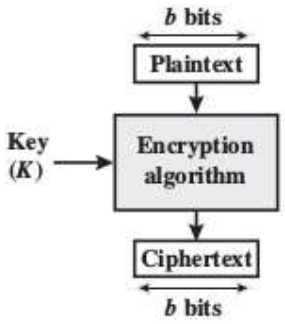
5. File Steganography: In this technique, a message is hidden in a file by altering the file structure. The hidden message can be revealed by analysing the file structure.

Here are some examples of steganography:

- Writing with invisible ink
- Embedding text in a picture (like an artist hiding their initials in a painting they've done)
- Backward masking a message in an audio file (remember those stories of evil messages recorded backward on rock and roll records?)
- Concealing information in either metadata or within a file header
- Hiding an image in a video, viewable only if the video is played at a particular frame rate
- Embedding a secret message in either the green, blue, or red channels of an RGB image

3. Block Ciphers:

A block cipher is a type of encryption algorithm that operates on fixed-length groups of bits, called blocks. Block ciphers are the elementary building blocks of many cryptographic protocols. They are ubiquitous in the storage and exchange of data, where such data is secured and authenticated via encryption. A block cipher uses blocks as an unvarying transformation. Even a secure block cipher is suitable for the encryption of only a single block of data at a time, using a fixed key. A multitude of modes of operation have been designed to allow their repeated use in a secure way to achieve the security goals of confidentiality and authenticity. However, block ciphers may also feature as building blocks in other cryptographic protocols, such as universal hash functions and pseudorandom number generators. There are several types of block ciphers, including the Data Encryption Standard (DES), the Advanced Encryption Standard (AES), and the Blowfish cipher.



3.1. Traditional Block Cipher Structure

Traditional block ciphers follow a specific structure that involves the use of a block of data and a key to perform encryption and decryption operations. The most common block cipher structures include the Feistel Network and the Substitution-Permutation Network (SPN).

Key Concept: Block ciphers are symmetric encryption algorithms that encrypt data in fixed-size blocks, typically 64 or 128 bits.
Same Key for Encryption and Decryption (Symmetric Encryption): They use the same key for both encryption and decryption.
Structure of a Traditional Block Cipher:

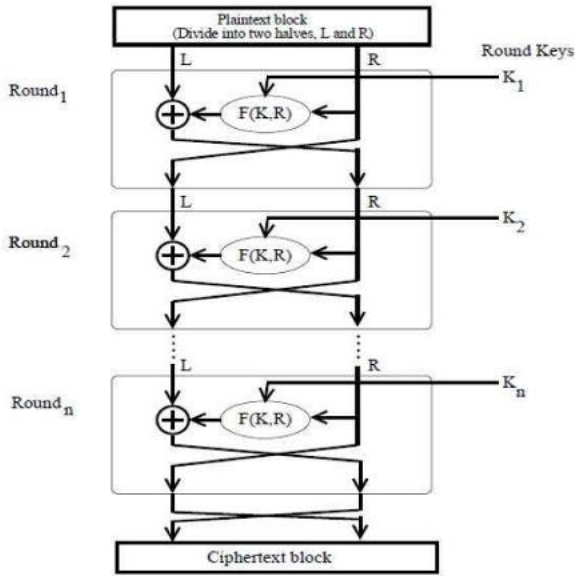
1. Plaintext Input:

The plaintext message is divided into blocks of the appropriate size.
2. Key Schedule:

The secret key is expanded into multiple subkeys, each used in a different round of encryption.
3. Round Function:

The core of a block cipher is the round function, which applies a series of transformations to the plaintext block.
It involves substitution, permutation, and XOR operations.
The round function is repeated multiple times (typically 10-16 rounds) to achieve strong encryption.
4. Feistel Structure (Common Design):

Many block ciphers use a Feistel structure, named after Horst Feistel.



The plaintext block is divided into two halves, L0 and R0.
In each round:

- A substitution is performed on one half (usually the right half).
- The result is XORed with the other half.
- The two halves are then swapped.

5. Ciphertext Output:

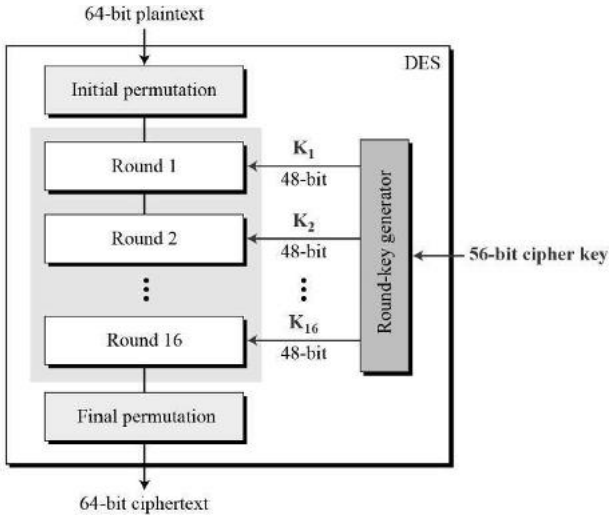
After the final round, the two halves are combined to form the ciphertext block.

3.2. The Data Encryption Standard (DES)

The Data Encryption Standard (DES) is a symmetric-key block cipher that was developed in the early 1970s and standardized in 1977. It was widely used for decades to protect sensitive electronic data, but is now considered outdated due to its short key length. However, it remains an important milestone in the history of cryptography and serves as a foundation for understanding modern ciphers.

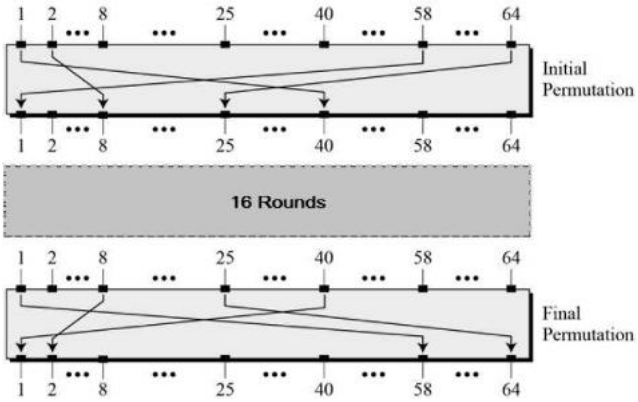
Key Features of DES:

- **Block Size**: 64 bits, meaning it encrypts data in blocks of 64 bits at a time.
- **Key Length**: 56 bits (effectively), although the actual key is 64 bits long, 8 bits are used for parity checking and discarded.
- **Feistel Structure**: DES employs a Feistel network, a common structure in block ciphers, which involves splitting the plaintext block into two halves and processing them through multiple rounds.
- **16 Rounds**: The encryption process involves 16 rounds of transformations, each round using a different 48-bit subkey derived from the main key.



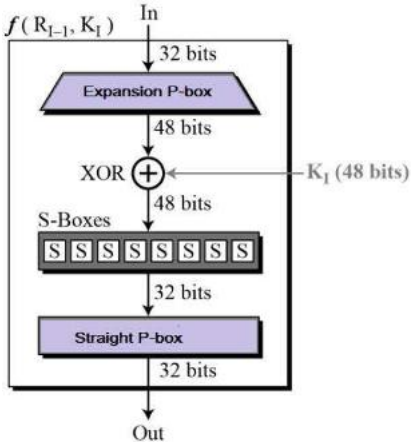
DES Encryption Process:

1. Initial Permutation and Final Permutation: The 64-bit plaintext block is shuffled using a fixed permutation table.



After the 16 rounds, the halves are combined and subjected to a final permutation, resulting in the 64-bit ciphertext block.

2. 16 Rounds of Encryption:
   Each round involves:



   a. **Expansion**: The 32-bit right half is expanded to 48 bits using a permutation table.
      - The expansion process takes the 32-bit right half of the data block as its input.
      - It employs a fixed permutation table, called the E-box, to rearrange the bits in a specific, non-linear pattern.

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

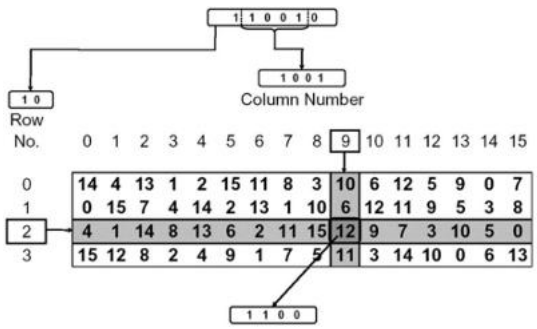
      - During this permutation, some bits are duplicated, effectively increasing the size from 32 bits to 48 bits.

      Example: If the 32-bit input is 10110101111000100100110111011001, the expanded 48-bit output would be 111111000110011111111110110101010101010101010.

   b. **Key Mixing**: The expanded right half is XORed with a 48-bit subkey.
      The key mixing process takes two inputs:
      - Expanded Right Half: The 48-bit result of the expansion process from the 32-bit right half of the data block.
      - Subkey: A 48-bit subkey derived from the main DES key for the current round.
      - Each bit of the expanded right half is XORed with the corresponding bit of the subkey. This exclusive-or operation combines the data and key bit values, resulting in a 48-bit mixed value.

   c. **Substitution**: The result is divided into 8 6-bit blocks, each passed through a substitution box (S-box) for non-linear substitution.
      - There are eight S-boxes in DES, each taking a 6-bit input and producing a 4-bit output.
      - The 48 bits from the Feistel function are divided into eight 6-bit blocks, and each block is processed by a separate S-box.
      - Each S-box performs a substitution based on its specific 4x16 table, replacing the 6 input bits with 4 output bits.

      Example of an S-box:



d. **Permutation**: The 32 bits from the S-boxes are rearranged using a permutation table.

Permutation Function P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

e. **XOR with Left Half**: The result is XORed with the 32-bit left half.
f. **Swap Halves**: The two halves are swapped for the next round.

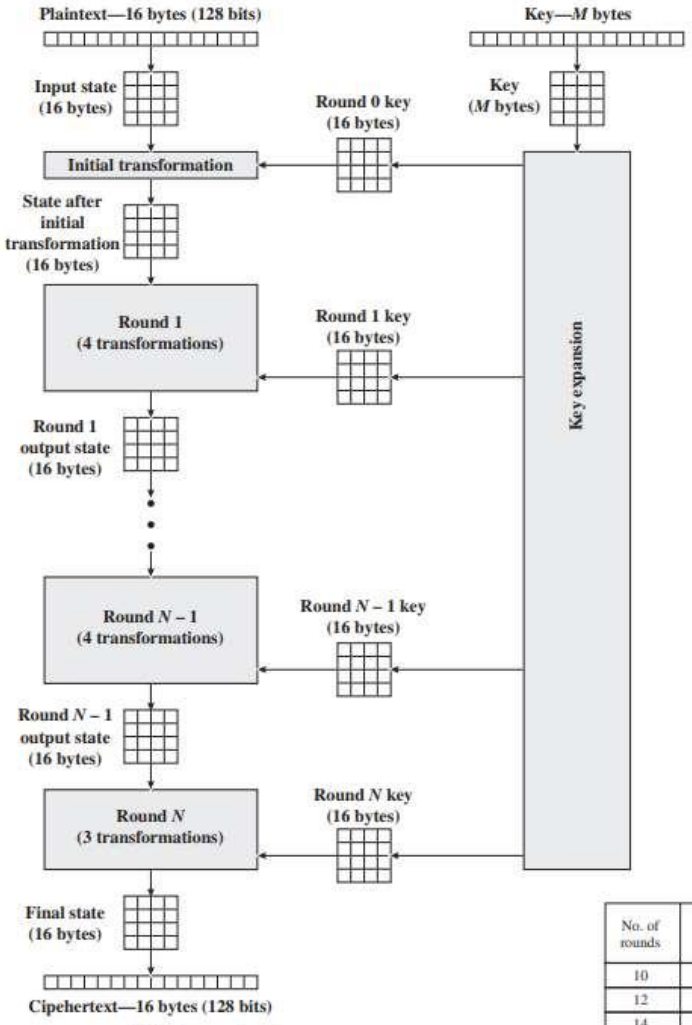
Security Concerns and Replacement:

- **Short Key Length**: The 56-bit key length of DES became vulnerable to brute-force attacks with increasing computing power.
- **Triple DES**: To address this, Triple DES was developed, which involves applying DES three times with three different keys.
- **AES**: Eventually, DES was officially replaced by the Advanced Encryption Standard (AES) in 2001, which offers stronger security with longer keys and more complex rounds.

4. Advanced Encryption Standard:

The Advanced Encryption Standard (AES) is a symmetric encryption algorithm established by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES is widely used for securing sensitive information and is considered a standard for cryptographic security.

AES Structure

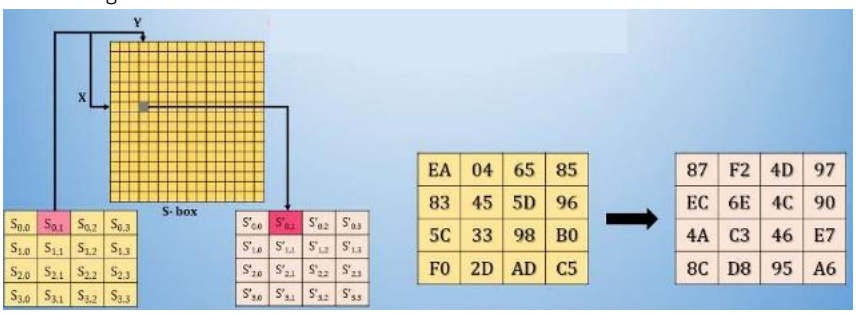


AES Encryption Process

No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Each round in the AES encryption process consists of the following operations:

SubBytes: Substitution of each byte in the block using a fixed S-box (Substitution box). The S-box introduces non-linearity into the algorithm.



ShiftRows: Shifting the rows of the block to create diffusion. The number of positions each byte is shifted depends on the row index.
Here is an example of how the ShiftRows step works:

	0	1	2	3
0	a	b	c	d
1	e	f	g	h
2	i	j	k	l
3	m	n	o	p

After the ShiftRows step, the matrix becomes:

	0	1	2	3
0	a	b	c	d
1	f	g	h	e
2	k	l	i	j
3	p	m	n	o

MixColumns: Transformation that mixes data in the columns. Each column is treated as a polynomial, multiplied by a fixed polynomial, and reduced modulo a predefined polynomial.

AddRoundKey: XORing the block with the round key generated during the key expansion step. Each byte of the block is XORed with the corresponding byte of the round key.

SubBytes	$b_{i,j} = S[a_{i,j}]$
ShiftRows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
MixColumns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

UNIT – II

1. Number Theory:

Number theory explores the properties and relationships of integers, playing a crucial role in cryptography, coding theory, and other mathematical fields.

1.1. The Euclidean Algorithm

The Euclidean Algorithm is an efficient method for computing the greatest common divisor (GCD) of two integers. The algorithm is based on the principle that the GCD of two numbers is the same as the GCD of the smaller number and the remainder of the larger number divided by the smaller number. Here is how the algorithm works:

If A = 0 then GCD (A, B) = B, since the GCD (0, B) = B, and we can stop.
If B = 0 then GCD (A, B) = A, since the GCD(A,0) = A, and we can stop.
Write A in quotient remainder form (A = B * Q + R)
Find GCD (B, R) using the Euclidean Algorithm since GCD (A, B) = GCD (B, R)
Here is an example of how to use the Euclidean Algorithm to find the GCD of 270 and 192:

A = 270, B = 192
270 = 192 * 1 + 78
192 = 78 * 2 + 36
78 = 36 * 2 + 6
36 = 6 * 6 + 0

Therefore, the GCD of 270 and 192 is 6.

The **Extended Euclidean Algorithm** is an algorithm to compute integers x and y such that ax + by = gcd(a,b) given a and b are known. The existence of such integers is guaranteed by Bézout's lemma. The extended Euclidean algorithm can be viewed as the reciprocal of modular exponentiation. By reversing the steps in the Euclidean algorithm, it is possible to find these integers x and y. The whole idea is to start with the GCD and recursively work our way backwards. This can be done by treating the numbers as variables until we end up with an expression that is a linear combination of our initial numbers. Here is how the algorithm works:

1. If b = 0, then x = 1 and y = 0, and gcd (a, b) = a.
2. If a = 0, then x = 0 and y = 1, and gcd(a,b) = b.
3. Otherwise, let q be the quotient when a is divided by b, and let r be the remainder. Then we have a = bq + r and gcd(a,b) = gcd(b,r). By the recursive call, we can find x1 and y1 such that bx1 + ry1 = gcd(b,r). Solving for x and y, we get x = y1 and y = x1 - y1 * q.

Here is an example of how to use the Extended Euclidean Algorithm to find the integers x and y such that 35x + 15y = 5:

```
a = 35, b = 15
35 = 15 * 2 + 5
15 = 5 * 3 + 0

gcd(35, 15) = 5
x = 1, y = -2
```

Therefore, the integers x and y such that 35x + 15y = 5 are x = 1 and y = -2.

1.2. Modular Arithmetic

Understanding Modular Arithmetic: The Clock of Numbers

Imagine a clock with 12 hours. It revolves around a cycle of 12, where numbers "wrap around" after reaching 12. Modular arithmetic follows this same principle, but with any chosen modulus (the number representing the cycle).

Key Concepts:

- **Congruence**: We say two numbers are congruent modulo m if they have the same remainder when divided by m. Notation: a ≡ b (mod m).
- **Residue**: The remainder when a number is divided by the modulus.
- **Modulo Operation**: The symbol "%" represents the modulo operation in programming. a % m gives the remainder when a is divided by m.
  Examples:

  5 ≡ 17 (mod 12) because both have a remainder of 5 when divided by 12.
  30 % 4 = 2 because 30 divided by 4 leaves a remainder of 2.
- **Basic Operations:**

  Addition: (a + b) % m ≡ (a % m + b % m) % m
  Example: (5 + 9) % 12 ≡ (14 % 12) ≡ 2
  Subtraction: (a - b) % m ≡ (a % m - b % m + m) % m
  Example: (15 - 8) % 12 ≡ (7 % 12 + 12) % 12 ≡ 7
  Multiplication: (a * b) % m ≡ (a % m * b % m) % m
  Example: (6 * 4) % 12 ≡ (6 % 12 * 4 % 12) % 12 ≡ 0

- **Properties**:

  Commutative: a + b ≡ b + a (mod m) and a * b ≡ b * a (mod m)
  Associative: (a + b) + c ≡ a + (b + c) (mod m) and (a * b) * c ≡ a * (b * c) (mod m)
  Distributive: a * (b + c) ≡ (a * b) + (a * c) (mod m)

- **Applications:**
  - **Cryptography**: Modular arithmetic is fundamental to many encryption algorithms, including RSA and Diffie-Hellman.
  - **Error detection and correction**: It's used in checksums and cyclic redundancy checks to detect and correct errors in data transmission.
  - **Computer science**: It's used in hash tables, random number generation, and other algorithms.

1.3. Fermat's and Euler's Theorems

Fermat's Little Theorem states that if p is a prime number and a is an integer not divisible by p, then a^(p-1) mod p = 1. For example, if p = 7 and a = 2, then 2^6 mod 7 = 1. Fermat's Little Theorem is used in many areas of mathematics, including number theory and cryptography.

In simpler terms, if you raise any integer not divisible by a prime number to the power of one less than that prime number, and then divide by the prime, the remainder will always be 1.

Example:

If p = 7 and a = 3, then:

a^(p-1) = 3^(7-1) = 3^6 = 729

729 ÷ 7 = 104 with a remainder of 1

Therefore, 3^(7-1) ≡ 1 (mod 7)

Key Points:

It applies only to prime numbers p.
a must not be divisible by p.
It concerns congruences modulo p (remainders when divided by p).
Applications:

Primality testing: Used to check if a number is likely prime. If a number fails Fermat's Little Theorem for multiple bases, it's certainly composite (not prime).
Cryptography: Underlying the RSA algorithm, essential for secure data transmission.

Euler's Totient Function is a function that counts the number of positive integers less than or equal to n that are relatively prime to n. It is denoted by the symbol ϕ(n). For example, ϕ(8) = 4 because the numbers 1, 3, 5, and 7 are relatively prime to 8. Euler's Totient Function is used in many areas of mathematics, including number theory and cryptography.

Euler's Theorem is a generalization of Fermat's Little Theorem. It states that if a and m are two positive integers that are relatively prime, then a^ϕ(m) mod m = 1. For example, if a = 2 and m = 7, then ϕ(7) = 6 and 2^6 mod 7 = 1. Euler's Theorem is used in many areas of mathematics, including number theory and cryptography.

Here is an example of how to use Euler's Theorem to find the remainder when 2^100 is divided by 7:

ϕ(7) = 6
2^6 mod 7 = 1
2^100 mod 7 = (2^6)^16 * 2^4 mod 7 = 1^16 * 16 mod 7 = 2

Therefore, the remainder when 2^100 is divided by 7 is 2.
Applications:

- Cryptography: Essential in RSA, Diffie-Hellman, and other cryptosystems.
- Number theory: Contributes to understanding prime number distribution and other concepts.
- Discrete mathematics: Used in counting problems and generating pseudorandom numbers.

1.4. The Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) is a mathematical theorem that provides a solution to a system of simultaneous linear congruences. It is named the Chinese Remainder Theorem because it was independently discovered by Chinese mathematicians Sunzi (circa 3rd century) and later by Qin Jiushao (13th century). The theorem is applicable in modular arithmetic and has various practical applications, especially in computer science and cryptography.

The Theorem:

Given integers a, b, m, n, where m and n are coprime (have no common factor other than 1), and congruences:
x ≡ a (mod m)
x ≡ b (mod n)
There exists a unique solution x modulo mn such that both congruences are satisfied.

Understanding the Concept: Think of dividing by 3, 5, and 7 as separate buckets. The theorem helps us find a number that leaves a specific remainder in each bucket while fitting within a larger container (mn).

Solving the Riddle:
Identify the congruences from the problem:

x ≡ 2 (mod 3)
x ≡ 3 (mod 5)
x ≡ 2 (mod 7)
Check if the moduli are coprime: 3 and 5 are, 7 and 3 are, and 7 and 5 are. All pairs are coprime!

Calculate the moduli product: mn = 3 * 5 * 7 = 105.

Find the modular inverses:

m' = n^-1 (mod m)
n' = m^-1 (mod n)
For this example, 2 (mod 5) and 5 (mod 3) satisfy these conditions.
Apply the CRT formula:

x ≡ a * n' * n + b * m' * m (mod mn)
In our case, x ≡ 2 * 5 * 5 + 3 * 2 * 3 (mod 105)

This simplifies to x ≡ 23 (mod 105)
Therefore, the number satisfying all the congruences is 23 (mod 105), which means there are 23 things in the riddle.

1.5. Discrete Logarithms

In mathematics, a discrete logarithm is an integer k that satisfies the equation b^k ≡ a (mod m) for given integers a, b, and m.

Defining the Problem: Given a base g, modulus p, and a residue b, the discrete logarithm problem (DLP) asks: What exponent x satisfies the congruence g^x ≡ b (mod p)?

In simpler terms:

Imagine a clock with p hours.
Raising g to a power is like repeatedly moving forward on the clock.
We want to find how many times we need to move forward (the exponent) to reach b.

Example:

If g = 3, p = 11, and b = 5, the DLP asks for the value of x that makes 3^x ≡ 5 (mod 11). The answer is x = 4, as 3^4 ≡ 81 ≡ 5 (mod 11).

Challenges and Applications:

- Computing discrete logarithms is computationally hard: Finding x for large p is considered intractable for current computers, making it a cornerstone of modern cryptography.
- Cryptography:
  - Diffie-Hellman key exchange: Enables secure communication over public channels.
  - ElGamal encryption: A public-key cryptosystem providing confidentiality.
- Other fields: Digital signatures, zero-knowledge proofs, pseudorandom number generation, and more.

2. Finite Fields:

A finite field, also known as a Galois field, is a mathematical structure that generalizes the properties of arithmetic operations (addition, subtraction, multiplication, and division) familiar from the field of real numbers to a finite set of elements. Finite fields are essential in various areas of mathematics, computer science, and engineering, including coding theory, cryptography, and error-correcting codes.

2.1. Finite Fields of the Form GF(p)

GF(p) is a set containing the integers from 0 to p-1, along with two operations:

- Addition (mod p): Add the numbers as usual, but if you exceed p-1, "wrap around" the result by subtracting p.
- Multiplication (mod p): Multiply the numbers as usual, but again, if you exceed p-1, "wrap around" by taking the remainder when dividing by p.

Example: In GF(5), adding 3 and 4 is not 7, but 2 (3 + 4 ≡ 2 mod 5). Similarly, multiplying 2 and 3 gives 1 (2 * 3 ≡ 1 mod 5).

Key Properties:

- Finite: There are only p elements in GF(p).
- Commutative: Both addition and multiplication are commutative (order doesn't matter).
- Associative: Both addition and multiplication are associative (grouping doesn't matter).
- Distributive: Multiplication distributes over addition.
- Field axioms: GF(p) satisfies all the axioms of a field, making it a complete and consistent system for arithmetic.

Applications:

- Cryptography: Used in error-correcting codes, elliptic curve cryptography, and other protocols.
- Coding theory: Plays a crucial role in designing reliable communication systems and detecting errors.
- Computer science: Found in polynomial representations, finite automata, and other algorithms.

2.2. Finite Fields of the Form GF(2^n)

Beyond the prime-based fields like GF(p), there exists another fascinating family of finite fields: those of the form GF(2^n), where 2^n represents a power of 2. These fields, often called Galois Fields or binary fields, offer unique properties and applications in various domains. Let's delve into their construction and characteristics:

Construction:

1. Start with GF(2): The smallest finite field, containing only 0 and 1, with operations performed modulo 2 (essentially addition and multiplication without carrying).
2. Construct extensions: To create GF(2^n), we employ special polynomials of degree n with coefficients in GF(2) called irreducible polynomials. These polynomials cannot be factored further, ensuring the resulting field has the desired structure.
3. Represent elements as polynomials: Elements in GF(2^n) are polynomials of degree less than n, with coefficients in GF(2).
4. Define arithmetic: Addition and multiplication are performed modulo the chosen irreducible polynomial, ensuring results remain within the field.

Key Properties:

- Finite: Contains 2^n elements.
- Commutative, associative, and distributive: Adheres to the same field axioms as GF(p).
- Characteristic 2: Adding any element to itself results in 0 (due to the underlying binary arithmetic).

Example:

GF(4) can be constructed using the irreducible polynomial x^2 + x + 1. Its elements are 0, 1, x, and x + 1.

Addition and multiplication are performed modulo this polynomial, leading to unique arithmetic rules within this field.

Applications:

- Error-correcting codes: Crucial in communication systems to detect and correct errors, ensuring reliable data transmission.
- Cryptography: Used in certain cryptographic algorithms, such as the Advanced Encryption Standard (AES).
- Coding theory, digital signal processing, and computational algebra: Find further applications in these fields.

3. Public Key Cryptography:

Public key cryptography is a cryptographic system that uses two keys, a public key and a private key, to encrypt and decrypt data. The public key is used to encrypt data, while the private key is used to decrypt data. The two keys are mathematically related, but it is computationally infeasible to determine the private key from the public key. Public key cryptography is also known as asymmetric cryptography because the two keys are not the same. Public key cryptography is widely used in many areas of computer science, including secure data transmission, digital signatures, and key exchange protocols such as the Diffie-Hellman key exchange.

3.1. Principles

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

1.) key distribution – how to have secure communications in general without having to trust a KDC with your key

2.) digital signatures – how to verify a message comes intact from the claimed sender

Public-key/two-key/asymmetric cryptography involves the use of two keys:

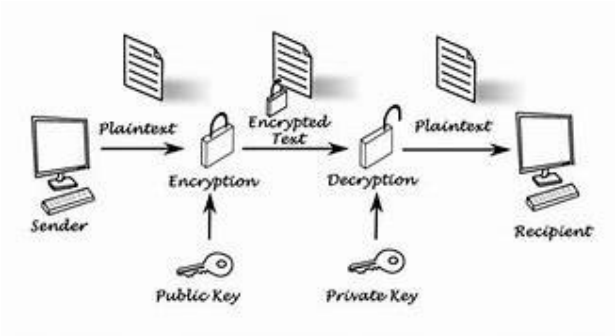
- a public-key, which may be known by anybody, and can be used to encrypt messages, and verify signatures
- a private-key, known only to the recipient, used to decrypt messages, and sign (create) signatures.

public-key cryptography is asymmetric because those who encrypt messages or verify signatures cannot decrypt messages or create signatures

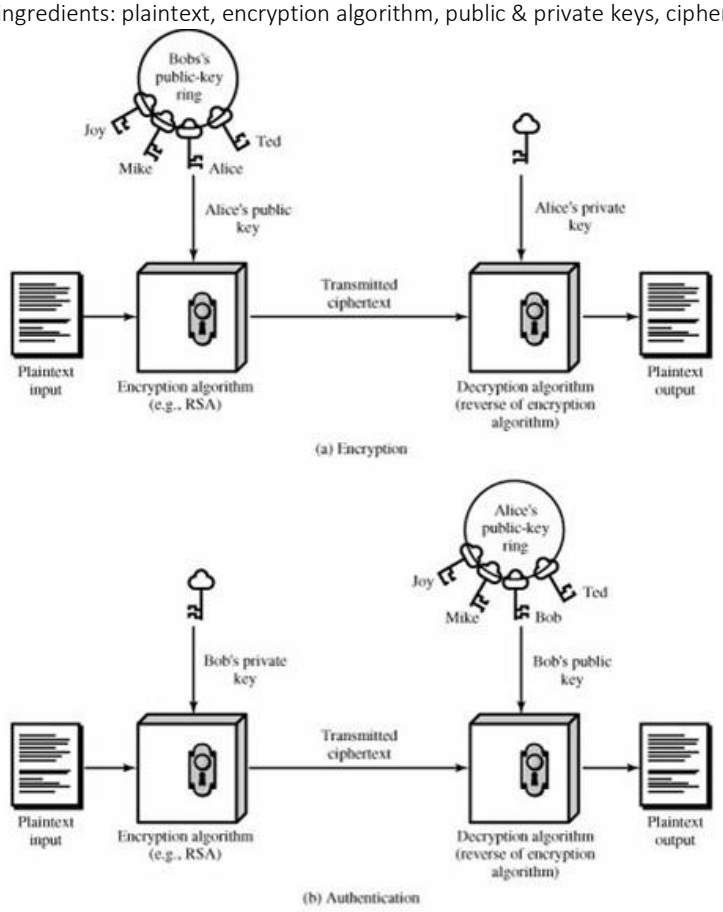
Public-Key algorithms rely on one key for encryption and different but related key for decryption. These algorithms have the following important characteristics:

- it is computationally infeasible to find decryption key knowing only algorithm & encryption key
- it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
- either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms like RSA)

The following figure illustrates public-key encryption process and shows that a public-key encryption scheme has six ingredients: plaintext, encryption algorithm, public & private keys, cipher text & decryption algorithm.



The following figure illustrates public-key encryption process and shows that a public-key encryption scheme has six ingredients: plaintext, encryption algorithm, public & private keys, cipher text & decryption algorithm.

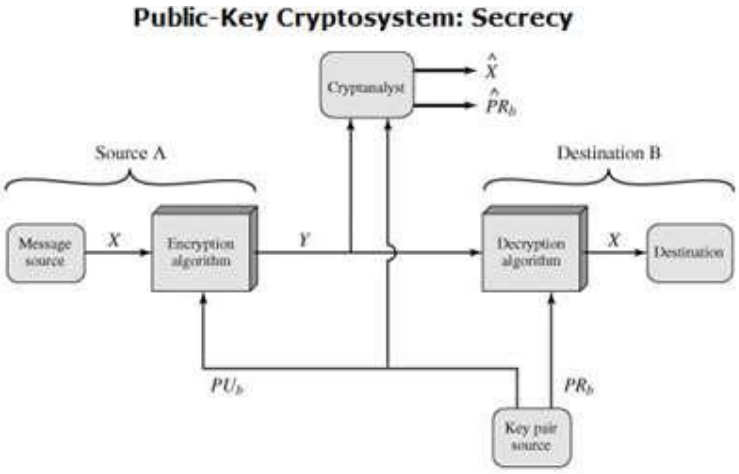


Encryption:

- Sender obtains recipient's public key.
- Encrypts message using recipient's public key.
- Only the recipient's private key can decrypt the message.

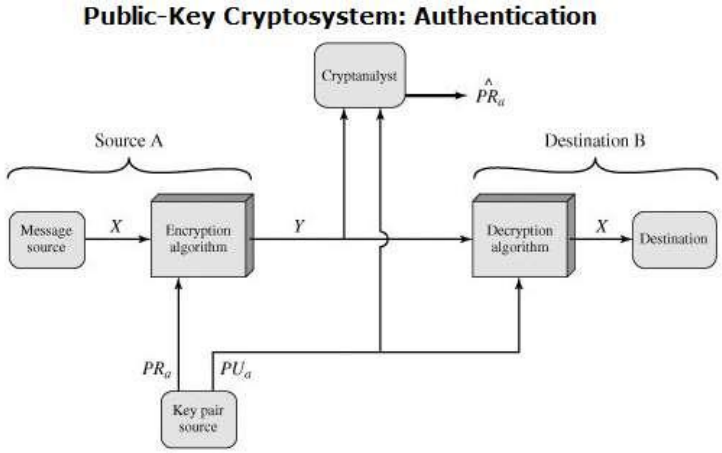
Decryption:

- Recipient receives encrypted message.
- Uses their private key to decrypt it.
- No one else can decrypt the message without the recipient's private key.

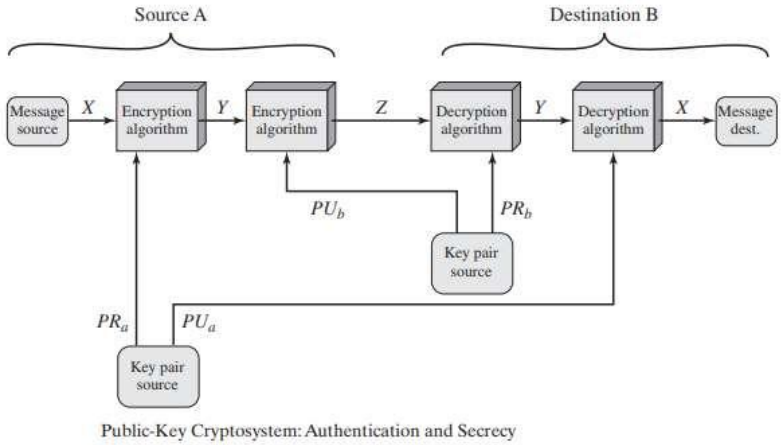


When a user wants to send a message, they use their private key to create authentication for the message. The authentication is a cryptographic representation of the message that is unique to the sender and the specific content of the message.

The recipient uses the sender's public key to verify the authentication. If the message is valid, it confirms that the message was authenticated by the holder of the private key associated with the public key.



It is to provide both the authentication function and confidentiality by a double use of the public-key



Public-Key Cryptosystem: Authentication and Secrecy

Applications for Public-Key Cryptosystems

• Encryption /decryption: The sender encrypts a message with the recipient's public key.

• Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

• Key exchange: Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

3.2. Public Key Cryptography Algorithms

3.2.1. RSA Algorithm

RSA algorithm was developed by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978.

The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n.

A typical size for n is 1024 bits, or 309 decimal digits.

That is, n is less than 2^{1024}

Description of the Algorithm:

RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to $\log_2(n) + 1$;

Figure below summarizes the RSA algorithm

Key Generation Alice

Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	gcd $(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key

Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key

Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example:

The keys were generated as follows

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17 × 11 = 187.
3. Calculate f(n) = (p - 1)(q - 1) = 16 × 10 = 160.
4. Select e such that e is relatively prime to f(n) = 160 and less than f(n); we choose e = 7.
5. Determine d such that de K 1 (mod 160) and d < 160.The correct value is d = 23,
   because 23 × 7 = 161 = (1 × 160) + 1;
   d can be calculated using the extended Euclid's algorithm

The resulting keys are public key PU = {7, 187} and private key PR = {23, 187}.

The example shows the use of these keys for a plaintext input of M = 88. For encryption, we need to calculate C = 887 mod 187. Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate M = 11^{23} mod 187:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

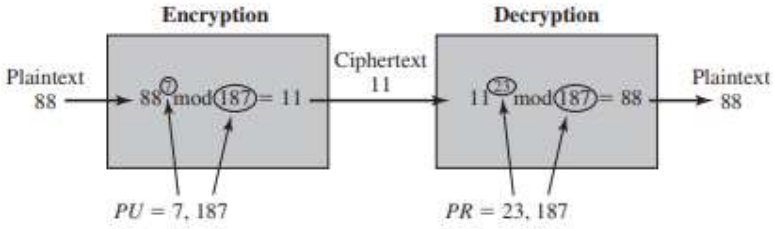
$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$



Example of RSA Algorithm

3.2.2. Diffie Hellman Key Exchange

The Diffie-Hellman Key Exchange Algorithm is a cryptographic method that enables two parties to agree on a shared secret key without any prior confidential information shared between them. It is widely used in many protocols like SSL/TLS, SSH, IPSec, and PKI.

Figure below summarizes the Diffie-Hellman key exchange algorithm.

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{XA} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{XB} \bmod q$

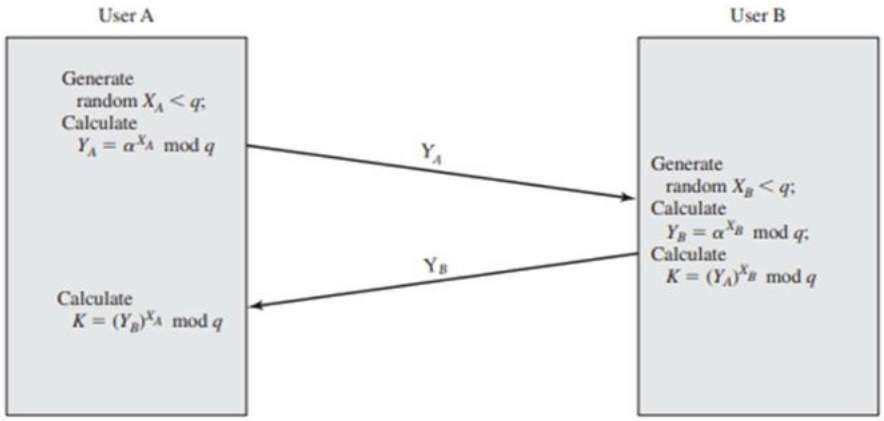
Calculation of Secret Key by User A

$$K = (Y_B)^{XA} \bmod q$$

Calculation of Secret Key by User B

$$K = (Y_A)^{XB} \bmod q$$

Figure below shows Diffie-Hellman Key Exchange



These two calculations produce identical results:

$$\begin{aligned} K &= (Y_B)^{X_A} \bmod q \\ &= (\alpha^{X_B} \bmod q)^{X_A} \bmod q \\ &= (\alpha^{X_B})^{X_A} \bmod q \qquad \text{by the rules of modular arithmetic} \\ &= \alpha^{X_B X_A} \bmod q \\ &= (\alpha^{X_A})^{X_B} \bmod q \\ &= (\alpha^{X_A} \bmod q)^{X_B} \bmod q \\ &= (Y_A)^{X_B} \bmod q \end{aligned}$$

Example:

Key exchange is based on the use of the prime number and a primitive root of 353, in this case α=3.

A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively.

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

$$\text{A computes } K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160.$$
$$\text{B computes } K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160.$$

3.2.3. Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is a public-key encryption algorithm that uses the algebraic structure of elliptic curves over finite fields. It is a more efficient and secure alternative to traditional public-key algorithms like RSA. ECC is used in many applications, including blockchain, mobile devices, and smart cards.

It's based on the complex mathematical properties of elliptic curves.

Key Concepts:

1. Elliptic Curves:
   - Defined by equations of the form y^2 = x^3 + ax + b over a finite field (remember GF(p) and GF(2^n)?).
   - Visually, they resemble a twisted donut shape.
   - Points on the curve, along with a special point called the "point at infinity," form the basis for cryptographic operations.
2. Key Pairs:
   - Each user generates a public key (a point on the curve) and a private key (a randomly chosen integer).
   - The public key is shared, while the private key remains secret.
   - They are mathematically linked through point addition and scalar multiplication.

Encryption and Decryption:

1. Encryption:

Message is represented as a point on the curve.

Sender uses recipient's public key to perform point operations, resulting in a different point on the curve.

Encrypted message is the coordinates of this new point.

2. Decryption:

Recipient uses their private key to reverse the point operations, revealing the original message point.

Example:

- Alice's public key: Point P = (3, 5) on the curve.
- Bob wants to send "Hello" (represented as point M on the curve).
- Bob encrypts: Uses Alice's public key to perform operations, resulting in point C.
- Bob sends C to Alice.
- Alice decrypts: Uses her private key to reverse operations, recovering point M and the message "Hello."

Applications:

- Secure communication protocols (TLS/SSL, SSH)
- Digital signatures
- Cryptocurrency (Bitcoin, Ethereum)
- IoT security
- Blockchain technology

All Figures below shows ECC (Elliptic Curve Cryptography) Diffie-Hellman Key Exchange

Global Public Elements	
$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation	
Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation	
Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A
$K = n_A \times P_B$

Calculation of Secret Key by User B
$K = n_B \times P_A$

The basic operations in ECC involve point addition and point multiplication on an elliptic curve. The elliptic curve equation takes the form:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

Here, a and b are parameters of the curve, and p is a prime number. ECC operates over a finite field defined by the prime p.

The following steps outline the process of ECC encryption and decryption:

Key Generation:

1. **Curve Selection:**
   - Choose an elliptic curve E defined over a finite field with parameters a, b, and p.
2. **Base Point Selection:**
   - Choose a base point G on the curve. The order (n) of G should be a large prime.
3. **Private Key Generation:**
   - Choose a private key d randomly from the interval $[1, n-1]$.
4. **Public Key Calculation:**
   - Compute the public key $Q = d \cdot G$, where Q is a point on the curve.

Encryption:

1. **Choose Recipient's Public Key:**
   - Obtain the recipient's public key Q_R.
2. **Choose a Random Number:**
   - Select a random number k from $[1, n-1]$.
3. **Compute the Cipher Points:**
   - Calculate $C_1 = k \cdot G$ and $C_2 = P + k \cdot Q_R$, where P is the plaintext point.
4. **Send the Cipher Text:**
   - Send (C_1, C_2) as the ciphertext.

Decryption:

1. **Compute Shared Secret:**
   - Use the recipient's private key d_R to compute $S = d_R \cdot C_1$.
2. **Recover Plaintext Point:**
   - Compute $P = C_2 - S$.

CRYPTOGRAPHY AND NETWORK SECURITY
UNIT-III

Hash Functions:

A hash function H accepts a variable-length block of data *M* as input and produces a fixed-size hash value

$h = H(M)$

Principal object is data integrity

The kind of hash function needed for security applications is referred to as a **cryptographic hash function**

Cryptographic hash function

- An algorithm for which it is computationally infeasible to find either:

(a) a data object that maps to a pre-specified hash result (the one-way property)

(b) two data objects that map to the same hash result (the collision-free property)

Applications of cryptographic hash functions:

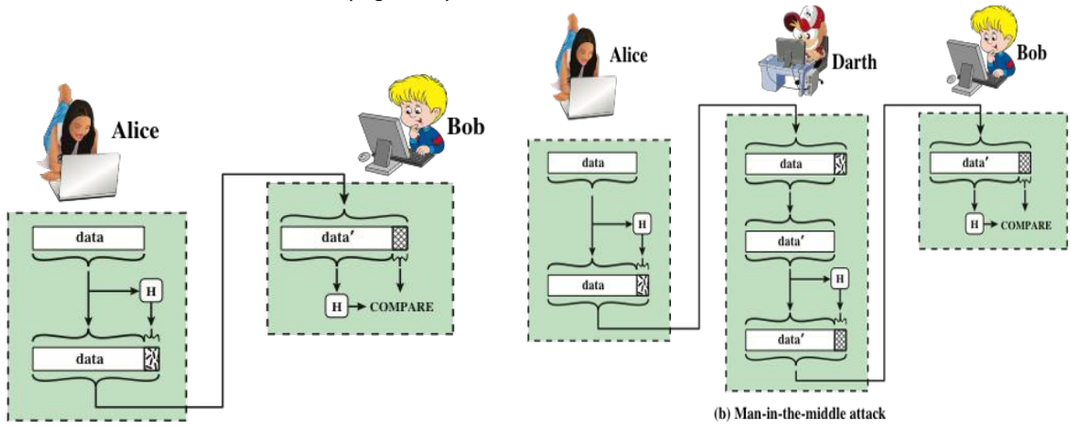
- Message Authentication
- Digital signature
- Other applications

Message Authentication:

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent (i.e., contain no modification, insertion, deletion, or replay). In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid. When a **hash function is used to provide message authentication**, the hash function value is often referred to as a **message digest.**

The essence of the use of a hash function for message authentication is as follows.

- The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message.
- The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value.
- If there is a mismatch, the receiver knows that the message (or possibly the hash value) has been altered (Figure a).



(a) Use of hash function to check data integrity

(b) Man-in-the-middle attack

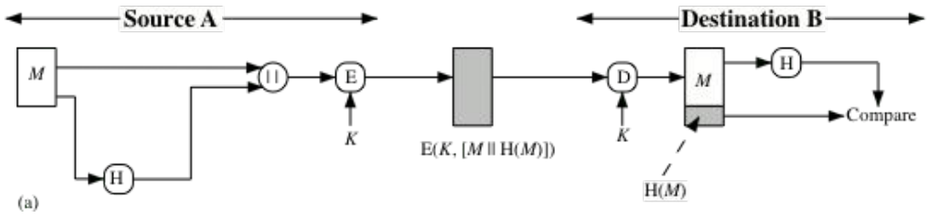
Figure 11.2 Attack Against Hash Function

The hash function must be transmitted in a secure fashion. That is, the hash function must be protected so that if an adversary alters or replaces the message, it is not feasible for adversary to also alter the hash value to fool the receiver. This type of attack is shown in Figure b. In this example, Alice transmits a data block and attaches a hash value. Darth intercepts the message, alters or replaces the data block, and calculates and attaches a new hash value. Bob receives the altered data with the new hash value and does not detect the change. To prevent this attack, the hash value generated by Alice must be protected.

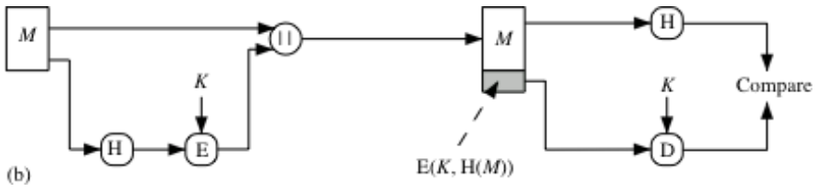
Ways Hash Code Can Be Used to Provide Message Authentication:

A 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. 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.

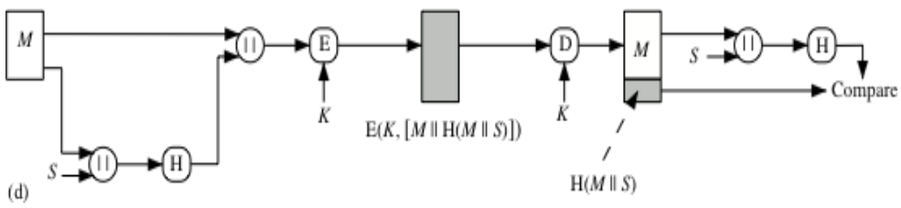


b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.



c) 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 recompute 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.

d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

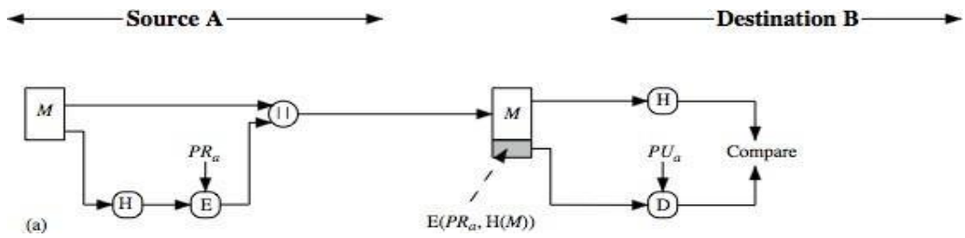


Digital Signatures:

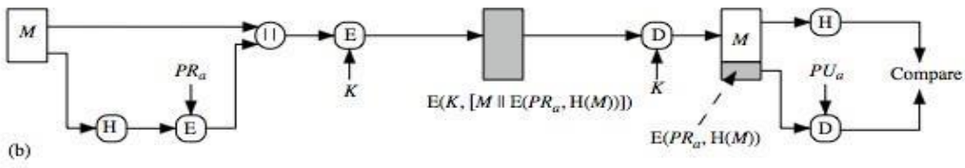
In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

In this case an attacker who wishes to alter the message would need to know the user's private key. There are **two types** how a hash code is used to provide a digital signature:

a. The hash code is encrypted, using public-key encryption and using the sender's private key. 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.



b) 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.



Other applications:

- to create a one-way password file that store hash of password not actual password
- for intrusion detection and virus detection it keep & check hash of files on system
- pseudorandom function (PRF) or pseudorandom number generator (PRNG).

Requirements and Security

Here there are two terms we need to define.
For a hash value h = H (x), we say that x is the **preimage of h**. That is, x is a data block whose hash function, using the function H, is h. Because H is a many-to-one mapping, for any given hash value h, there will in general be multiple preimages.
A **collision** occurs if we have x ≠ y and H (x) = H (y). Because we are using hash functions for data integrity, collisions are clearly undesirable.

Requirements for a Cryptographic Hash Function H:

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness

Table lists the generally accepted requirements for a cryptographic hash function.
The first three properties are requirements for the practical application of a hash function. The fourth property, **preimage (**for a hash value *h = H(x),* we say that x is the **preimage** of ***h***) **resistant**, is the one-way property: it is easy to generate a code given a message, but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value. The fifth property, **second preimage resistant**, guarantees that it is impossible to find an alternative message with the same hash value as a given message. This prevents forgery when an encrypted hash code is used. A hash function that satisfies the first five properties in is referred to as a **weak hash function**. If the sixth property, collision resistant, is also satisfied, then it is referred to as a **strong hash function**. A strong hash function protects against an attack in which one party generates a message for another party to sign. The final requirement, **pseudorandomness**, has not traditionally been listed as a requirement of cryptographic hash functions, but is more or less implied.

Attacks on hash functions:

As with encryption algorithms, there are two categories of attacks on hash functions:

1. brute-force attacks and
2. Cryptanalysis

- A **brute-force attack** does not depend on the specific algorithm but depends only on bit length. In the case of a hash function, a brute-force attack depends only on the bit length of the hash value.
- A **cryptanalysis**, in contrast, is an attack based on weaknesses in a particular cryptographic algorithm.

Birthday Attacks:

- ► For a collision resistant attack, an adversary wishes to find two messages or data blocks that yield the same hash function
  - The effort required is explained by a mathematical result referred to as the ***birthday paradox***
- ► How the birthday attack works:?
  - The source (A) is prepared to sign a legitimate message *x* by appending the appropriate *m*-bit hash code and encrypting that hash code with A's private key
  - Opponent generates $2^{m/2}$ variations *x'* of *x*, all with essentially the same meaning, and stores the messages and their hash values
  - Opponent generates a fraudulent message *y* for which A's signature is desired
  - Two sets of messages are compared to find a pair with the same hash
  - The opponent offers the valid variation to A for signature which can then be attached to the fraudulent variation for transmission to the intended recipient
    - Because the two variations have the same hash code, they will produce the same signature and the opponent is assured of success even though the encryption key is not known

Hash Function Cryptanalysis:

As with encryption algorithms, cryptanalytic attacks on hash functions seek to exploit some property of the algorithm to perform some attack other than an exhaustive search. In recent years, have much effort, and some successes, in developing cryptanalytic attacks on hash functions. Must consider the overall structure of a typical secure hash function, referred to as an iterated hash function.

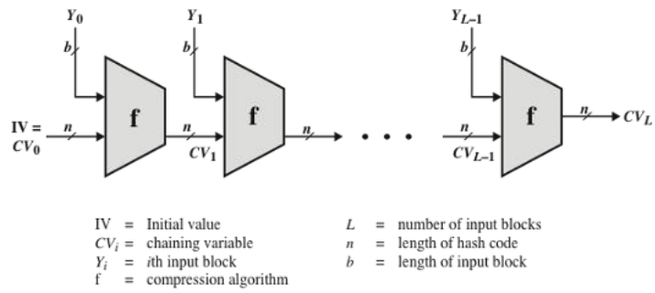


Figure 11.8 General Structure of Secure Hash Code

This was proposed by Merkle and is the structure of most hash functions in use today. The hash function takes an input message and partitions it into *L* fixed-sized blocks of *b* bits each. If necessary, the final block is padded to *b* bits. The final block also includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult. The hash algorithm involves repeated use of a **compression function**, *f*, that takes two inputs (an *n*-bit input from the previous step, called the chaining variable, and a *b*-bit block) and produces an *n*-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, $b > n$; hence the term compression. The motivation for this iterative structure stems from the observation by Merkle and Damgard that if the compression function is collision resistant, then so is the resultant iterated hash function. Therefore, the structure can be used to produce a secure hash function to operate on a message of any length. Cryptanalysis of hash functions focuses on the internal structure of f and is based on attempts to find efficient techniques for producing collisions for a single execution of f. Once that is done, the attack must take into account the fixed value of IV. The attack on f depends on exploiting its internal structure. The attacks that have been mounted on hash functions are rather complex.

Secure Hash Algorithm(SHA):

SHA was originally designed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.Was revised in 1995 as SHA-1. Based on the hash function MD4 and its design closely models MD4.Produces 160-bit hash values. In 2002 NIST produced a revised version of the standard that defined three new versions of SHA with hash value lengths of 256, 384, and 512.Collectively known as SHA-2.

Table Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

SHA-512 LOGIC:

The algorithm takes as input a message with a maximum length of less than 2^{128} bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

The processing consists of the following steps:

- **Step 1: Append padding bits**, the message is padded so that its length is congruent to 896 modulo 1024 [length = 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.
- **Step 2: Append length**. A block of 128 bits is appended to the message
- **Step 3: Initialize hash buffer**, A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h).
- **Step 4: Process the message in 1024-bit** (128-word) blocks, which forms the heart of the algorithm. This contains 80 rounds.
- **Step 5: Output the final state value as the resulting hash**

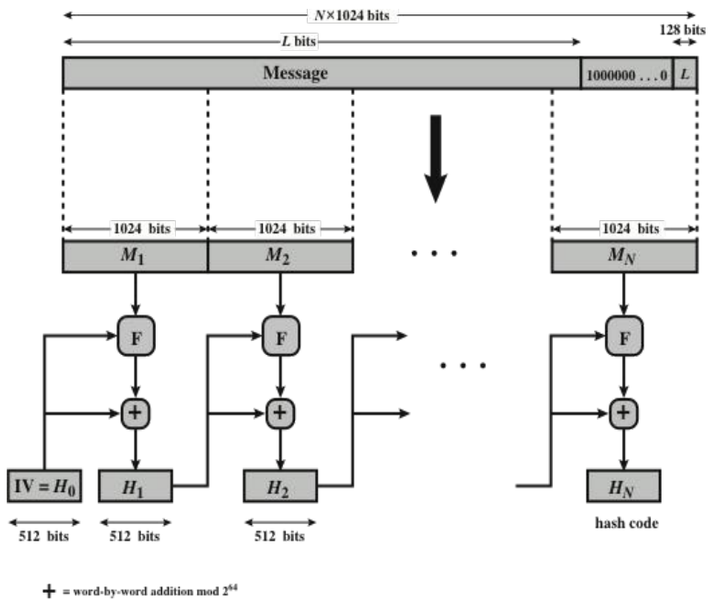


Figure 11.9 Message Digest Generation Using SHA-512

SHA-512 Compression Function:

- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
  - updating a 512-bit buffer
  - using a 64-bit value derived from the current message block
  - and a round constant based on cube root of first 80 prime numbers

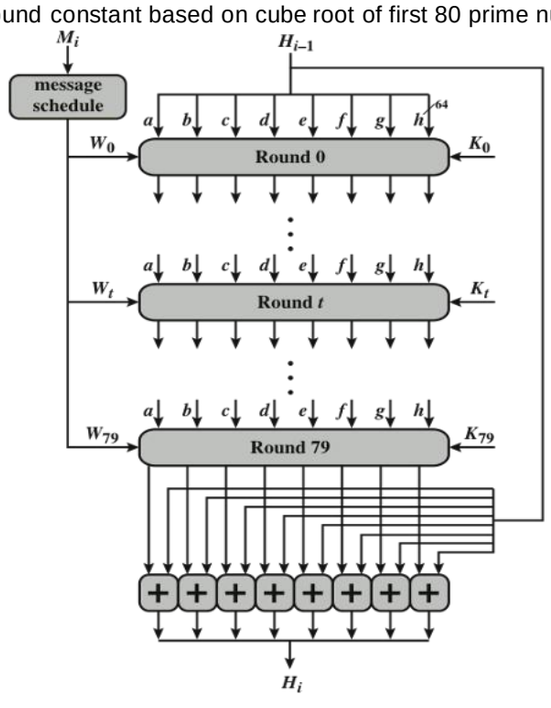


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

SHA-512 Round Function:

The structure of each of the 80 rounds is shown in Stallings Figure 11.10. Each 64-bit word is shuffled along one place, and in some cases manipulated using a series of simple logical functions (ANDs, NOTs, ORs, XORs, ROTates), in order to provide the avalanche & completeness properties of the hash function. The elements are:

Ch(e,f,g) = (e AND f) XOR (NOT e AND g)

Maj(a,b,c) = (a AND b) XOR (a AND c) XOR (b AND c)

∑(a) = ROTR(a,28) XOR ROTR(a,34) XOR ROTR(a,39)

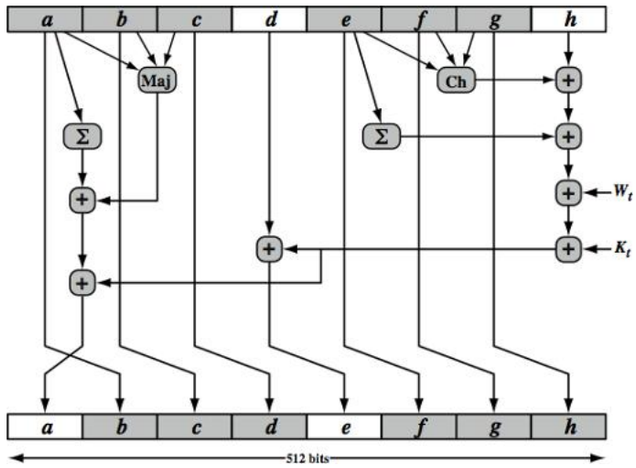
∑(e) = ROTR(e,14) XOR ROTR(e,18) XOR ROTR(e,41)

+ = addition modulo 2^64

Kt = a 64-bit additive constant

Wt = a 64-bit word derived from the current 512-bit input block.

Six of the eight words of the output of the round function involve simply permutation (*b, c, d, f, g, h*) by means of rotation. This is indicated by shading in Figure. Only two of the output words (*a, e)* are generated by substitution. Word e is a function of input variables *d, e, f, g, h,* as well as the round word W t and the constant Kt. Word a is a function of all of the input variables, as well as the round word W t and the constant Kt.



4.5. Message Authentication:

message authentication is concerned with: protecting the integrity of a message, validating identity of originator, non-repudiation of origin (dispute resolution)

Message Security Requirements

In the context of communications across a network, the following attacks can be identified.

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. **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.

3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This 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 nonreceipt by someone other than the message recipient.

4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. **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.

7. **Source repudiation:** Denial of transmission of message by source.

8. **Destination repudiation:** Denial of receipt of message by destination.

Message Authentication Functions:

There are three functions used:

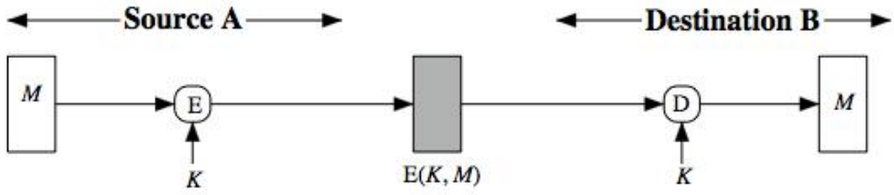
- ► **hash function**: A function that maps a message of any length into a fixed-length hash value which serves as the authenticator
- ► **message encryption**: The ciphertext of the entire message serves as its authenticator
- ► **message authentication code (MAC)**: A function of the message and a secret key that produces a fixed-length value that serves as the authenticator

Message Encryption:

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

Symmetric Message Encryption:

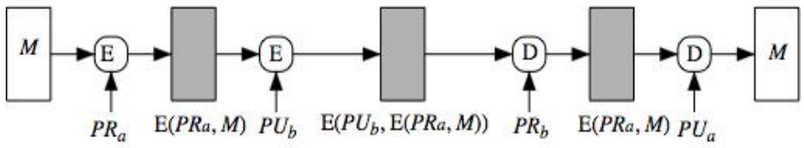
- encryption can also provide authentication
- if symmetric encryption is used then:
  - receiver know sender must have created it
  - since only sender and receiver now key used
  - know content cannot of been altered
  - if message has suitable structure, redundancy or a checksum to detect any changes



(a) Symmetric encryption: confidentiality and authentication

Public-Key Message Encryption:

- if public-key encryption is used:
  - encryption provides no confidence of sender
    - since anyone potentially knows public-key
  - however, if
    - sender **signs** message using their private-key
    - then encrypts with recipient's public key
    - have both secrecy and authentication
  - again need to recognize corrupted messages
  - but at cost of two public-key uses on message

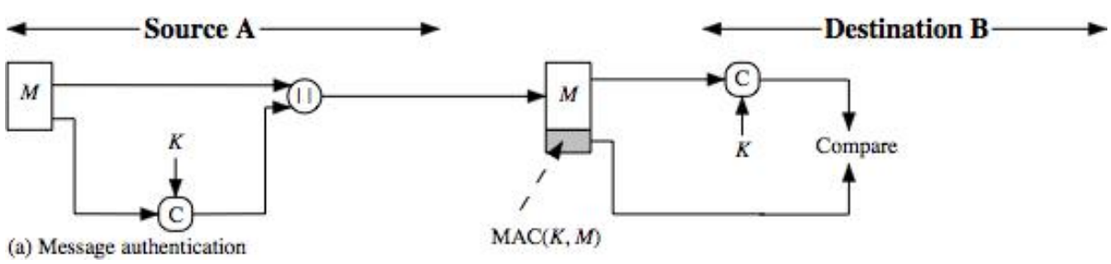


(d) Public-key encryption: confidentiality, authentication, and signature

Message Authentication Code (MAC):

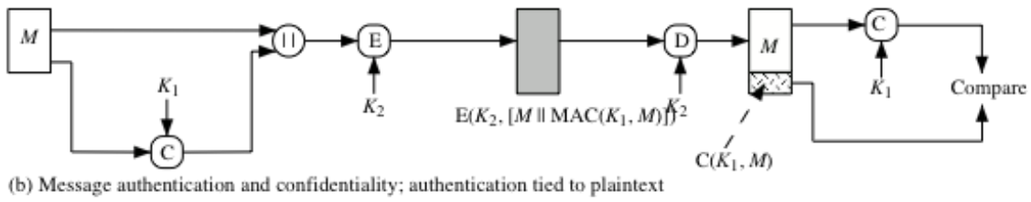
An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key: *MAC = C(K, M).*

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC Figure a. If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then the receiver is assured that the message has not been altered, is from the alleged sender, and if the message includes a sequence number then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number. A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many - to-one function. A MAC function is similar to encryption, except that the MAC algorithm need not be reversible, as it must for decryption.

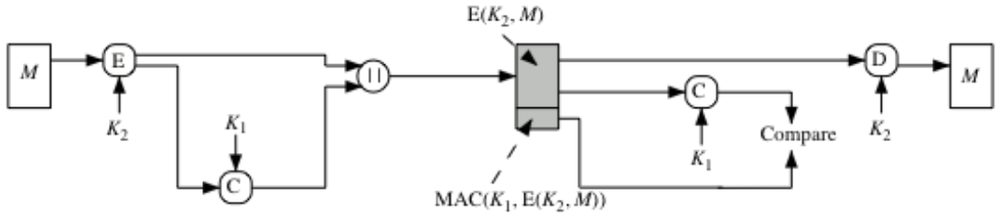


(a) Message authentication

The process depicted on the above provides authentication but not confidentiality, because the message as a whole is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (see Figure b) or before (see Figure c) the MAC algorithm. In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver. Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure 12.4b is used. Can use MAC in circumstances where just authentication is needed (or needs to be kept), see text for examples (e.g. such as when the same message is broadcast to a number of destinations; when one side has a heavy load and cannot afford the time to decrypt all incoming messages; or do not need to keep messages secret, but must authenticate messages). Finally, note that the MAC does not provide a digital signature because both sender and receiver share the same key.



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



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

Figure: Basic Uses of Message Authentication Code(MAC)

MAC Properties:

A MAC (also known as a cryptographic checksum, fixed-length authenticator, or tag) is generated by a function C. The MAC 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 MAC.

The MAC function is a many-to-one function, since potentially many arbitrarily long messages can be condensed to the same summary value, but don't want finding them to be easy.

HMAC Design Objectives:

RFC 2104 lists the following design objectives for HMAC:

- To use, without modifications, available hash functions. In particular, hash functions that perform well in software, and for which code is freely and widely available.
- To allow for easy replace ability of the embedded hash function in case faster or more secure hash functions are found or required.
- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mec hanism based on reasonable assumptions about the embedded hash function.

HMAC:

The idea of a keyed hash evolved into HMAC, designed to overcome some problems with the original proposals. It involves hashing padded versions of the key concatenated with the message, and then with another outer hash of the result prepended by another padded variant of the key. The hash function need only be used on 3 more blocks than when hashing just the original message (for the two keys + inner hash). HMAC can use any desired hash function, and has been shown to have the same security as the underlying hash function. Can choose the hash function to use based on speed/security concerns.

HMAC Overview:

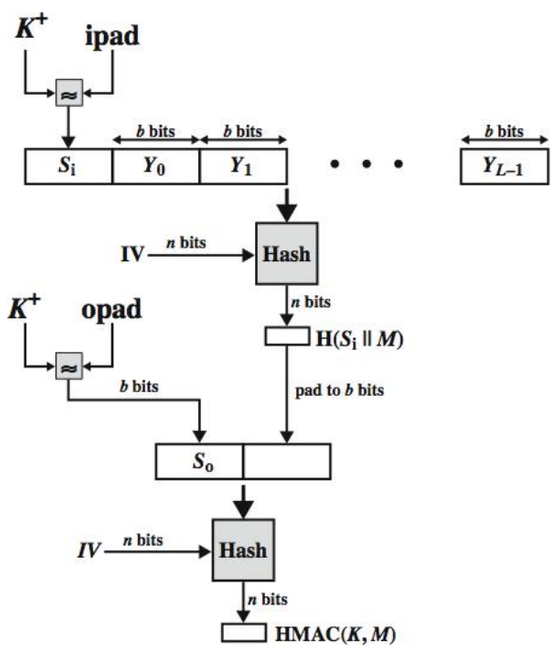


Figure illustrates the overall operation of HMAC:

$HMAC_K$ = Hash[(K^+ XOR opad) || Hash[(K^+ XOR ipad) || M)]

where:

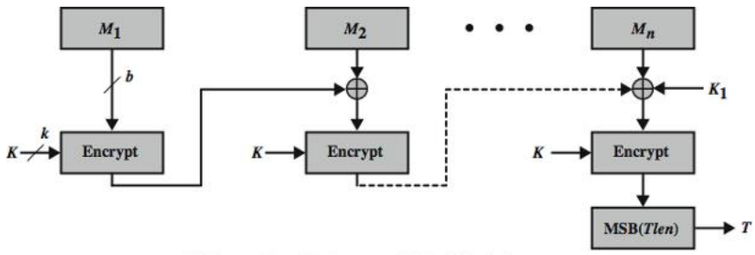
K^+ is K padded with zeros on the left so that the result is b bits in length

ipad is a pad value of 36 hex repeated to fill block
opad is a pad value of 5C hex repeated to fill block
M is the message input to HMAC (including the padding specified in the embedded hash function)
Note that the XOR with ipad results in flipping one-half of the bits of *K.* Similarly, the XOR with opad results in flipping one-half of the bits of K, but a different set of bits. In effect, pseudorandomly generated two keys from K. HMAC should execute in approximately the same time as the embedded hash function for long messages. HMAC adds three executions of the hash compression function (for *Si, So*, and the block produced from the inner hash). A more efficient implementation is possible by precomputing the internal hash function on (K^+ XOR opad) and (K^+ XOR ipad) and inserting the results into the hash processing at start & end. With this implementation, only one additional instance of the compression function is added to the processing normally produced by the hash function. This is especially worthwhile if most of the messages for which a MAC is computed are short.

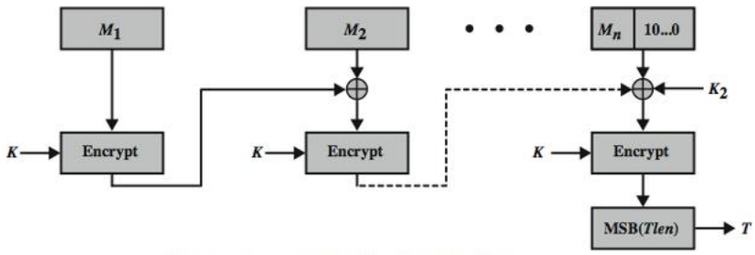
CMAC:

The Data Authentication Algorithm cipher-based MAC has been widely adopted in government and industry. Has been shown to be secure, with the following restriction. Only messages of one fixed length of *mn* bits are processed, where *n* is the cipher block size and *m* is a fixed positive integer. This limitation can be overcome using multiple keys, which can be derived from a single key. This refinement has been adopted by NIST as the cipher-based message authentication code (CMAC) mode of operation, for use with AES and triple DES. It is specified in NIST Special Publication 800-38B.

CMAC Overview:



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

4.9DIGITAL SIGNATURES

The most important development from the work on public-key cryptography is the **digital signature**. Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other either fraudulently creating, or denying creation, of a message. A digital signature is analogous to the handwritten signature, and provides a set of security capabilities that would be difficult to implement in any other way. It must have the following properties:

- It must verify the author and the date and time of the signature
- It must to authenticate the contents at the time of the signature
- It must be verifiable by third parties, to resolve disputes

Thus, the digital signature function includes the authentication function.

DIGITAL SIGNATURE MODEL

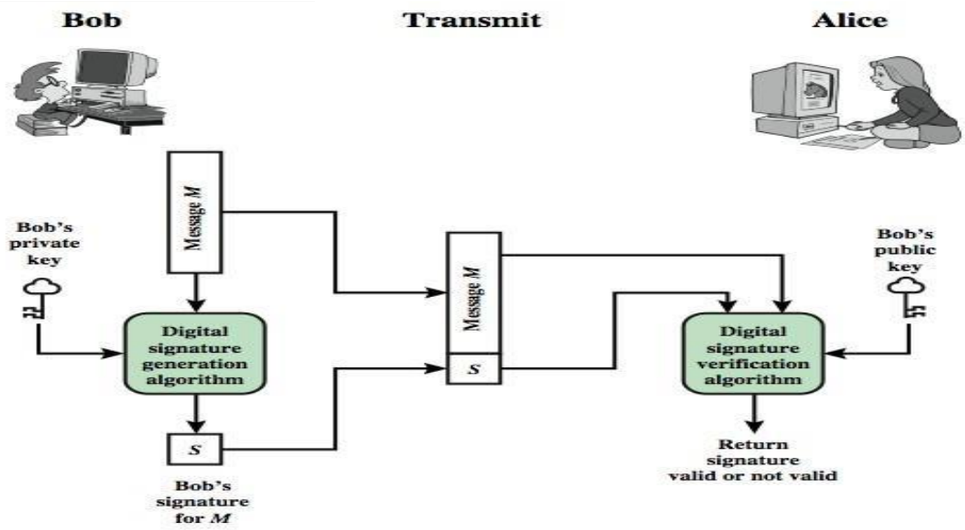


Figure is a generic model of the process of making and using digital signatures. Bob can sign a message using a digital signature generation algorithm. The inputs to the algorithm are the message and Bob's private key. Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key.

Attacks and Forgeries:

lists the following types of attacks, in order of increasing severity. Here A denotes the user whose signature is being attacked and C denotes the attacker.

- **Key-only attack**: C only knows A's public key.
- **Known message attack**: C is given access to a set of messages and signatures.
- **Generic chosen message attack**: C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic because it does not depend on A's public key; the same attack is used against everyone.
- **Directed chosen message attack**: Similar to the generic attack, except that the list of messages is chosen after C knows A's public key but before signatures are seen.
- **Adaptive chosen message attack**: C is allowed to use A as an "oracle." This means the A may request signatures of messages that depend on previously obtained message-signature pairs.

Then defines success as breaking a signature scheme as an outcome in which C can do any of the following with a non-negligible probability:

- **Total break**: C determines A's private key. • Universal forgery: C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.
- **Selective forgery**: C forges a signature for a particular message chosen by C.
- **Existential forgery**: C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.

Digital Signature Requirements:

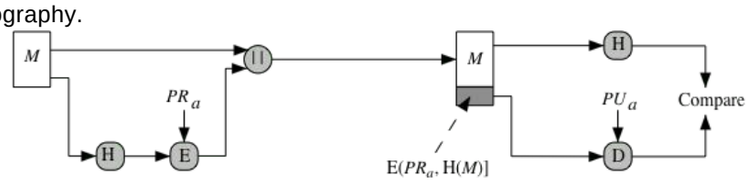
- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures:

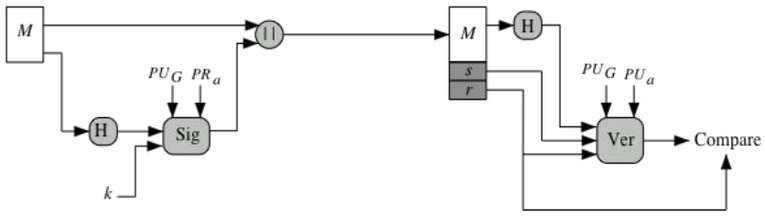
The term *direct digital* signature refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. Direct Digital Signatures involve the direct application of public-key algorithms involving only the communicating parties. A digital signature may be formed by encrypting the entire message with the sender's private key, or by encrypting a hash code of the message with the sender's private key. Confidentiality can be provided by further encrypting the entire message plus signature using either public or private key schemes. It is important to perform the signature function first and then an outer confidentiality function, since in case of dispute, some third party must view the message and its signature. But these approaches are dependent on the security of the sender's private-key. Will have problems if it is lost/stolen and signatures forged. The universally accepted technique for dealing with these threats is the use of a digital certificate and certificate authorities. Also need time-stamps and timely key revocation.

NIST Digital Signature Algorithm:

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Algorithm (DSA). The DSA makes use of the Secure Hash Algorithm (SHA). The DSA was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2, subsequently updated to FIPS 186-3 in 2009. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.



(a) RSA Approach



(b) DSA Approach

Figure 13.3 Two Approaches to Digital Signatures

The DSA uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

Above Figure contrasts the DSA approach for generating digital signatures to that used with RSA. In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

The DSA approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number k generated for this particular signature. The signature function also depends on the sender's private key (PR_a) and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (PU_G). The result is a signature consisting of two components, labeled s and r.

At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (PU_a), which is paired with the sender's private key. The output of the verification function is a value that is equal to the signature component r if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

The Digital Signature Algorithm:

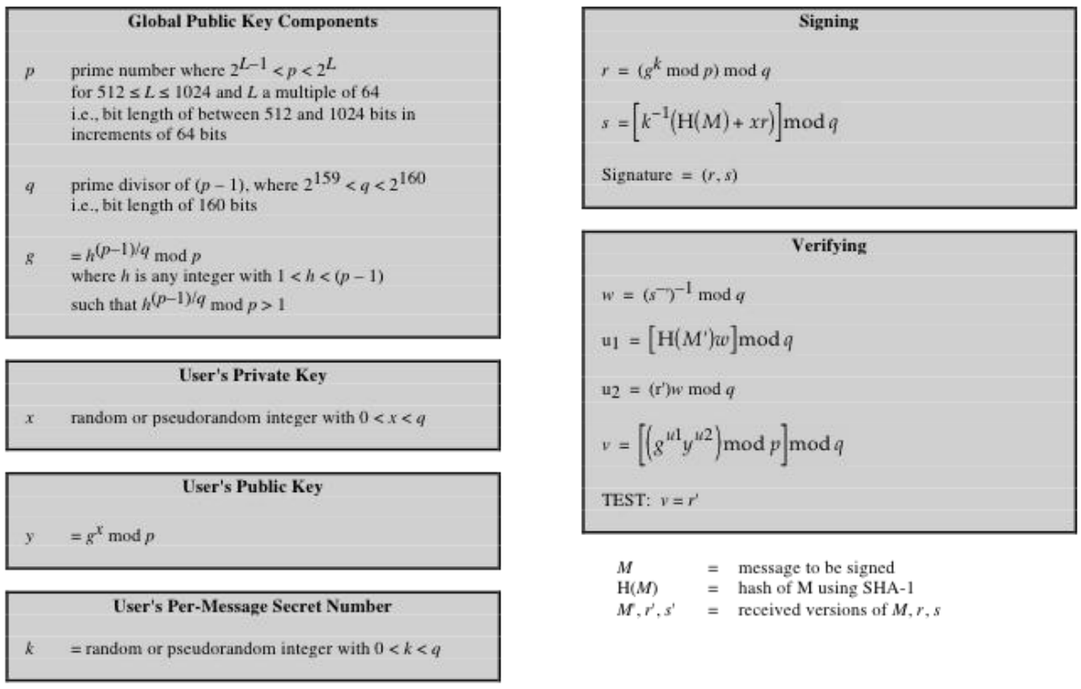


Figure 13.4 The Digital Signature Algorithm (DSS)

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by ElGamal and Schnorr. The DSA signature scheme has advantages, being both smaller (320 vs 1024bit) and faster (much of the computation is done modulo a 160 bit number), over RSA. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

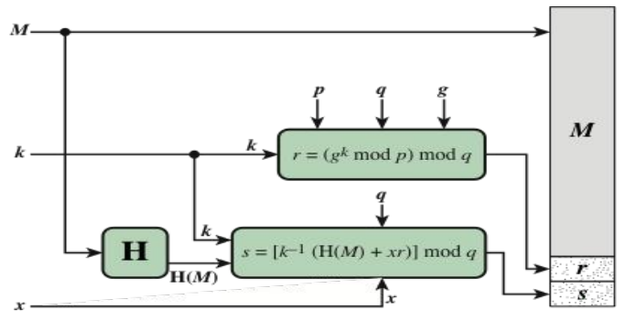
DSA typically uses a common set of global parameters (p,q,g) for a community of clients, as shown. A 160-bit prime number q is chosen. Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides (p – 1). Finally, g is chosen to be of the form $h^{(p-1)/q}$ mod p where h is an integer between 1 and (p – 1) with the restriction that g must be greater than 1. Thus, the global public key components of DSA have the same for as in the Schnorr signature scheme.

Then each DSA uses chooses a random private key x, and computes their public key as shown. The calculation of the public key y given x is relatively straightforward. However, given the public key y, it is computationally infeasible to determine x, which is the discrete logarithm of y to base g, mod p.

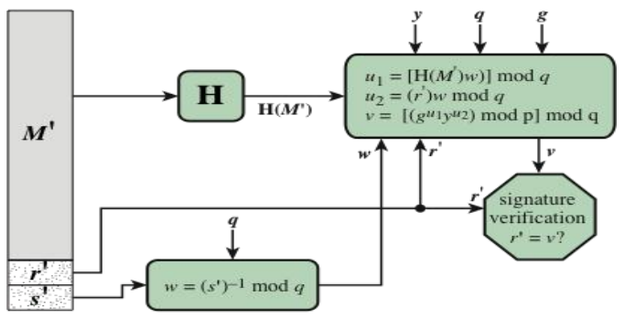
To create a signature, a user calculates two quantities, r and s, that are functions of the public key components (p,q,g), the user's private key (x), the hash code of the message H(M), and an additional integer k that should be generated randomly or pseudo-randomly and be unique for each signing. This is similar to ElGamal signatures, with the use of a per message temporary signature key k, but doing calculations first mod p, then mod q to reduce the size of the result. The signature (r,s) is then sent with the message to the recipient. Note that computing r only involves calculation mod p and does not depend on message, hence can be done in advance. Similarly with randomly choosing k's and computing their inverses.

At the receiving end, verification is performed using the formulas shown. The receiver generates a quantity v that is a function of the public key components, the sender's public key, and the hash of the incoming message. If this quantity matches the r component of the signature, then the signature is validated. Note that the difficulty of computing discrete logs is why it is infeasible for an opponent to recover k from r, or x from s. Note also that nearly all the calculations are mod q, and hence are much faster save for the last step.

The structure of this function is such that the receiver can recover r using the incoming message and signature, the public key of the user, and the global public key. It is c ertainly not obvious that such a scheme would work.



(a) Signing



(b) Verifying

Figure 13.5 DSA Signing and Verifying

Key management:

One of the major roles of public-key encryption is to address the problem of key distribution. There are actually two distinct aspects to the use of public-key encryption in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

Public-Key Certificates

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large. Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

The solution to this problem is the public-key certificate. In essence, a certificate consists of a public key plus a User ID of the key owner, with the whole block signed by a trusted third party. Typically, the third party is a certificate authority (CA) that is trusted by the user community, such as a government agency or a financial institution. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. Figure 3.12 illustrates the process.

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security

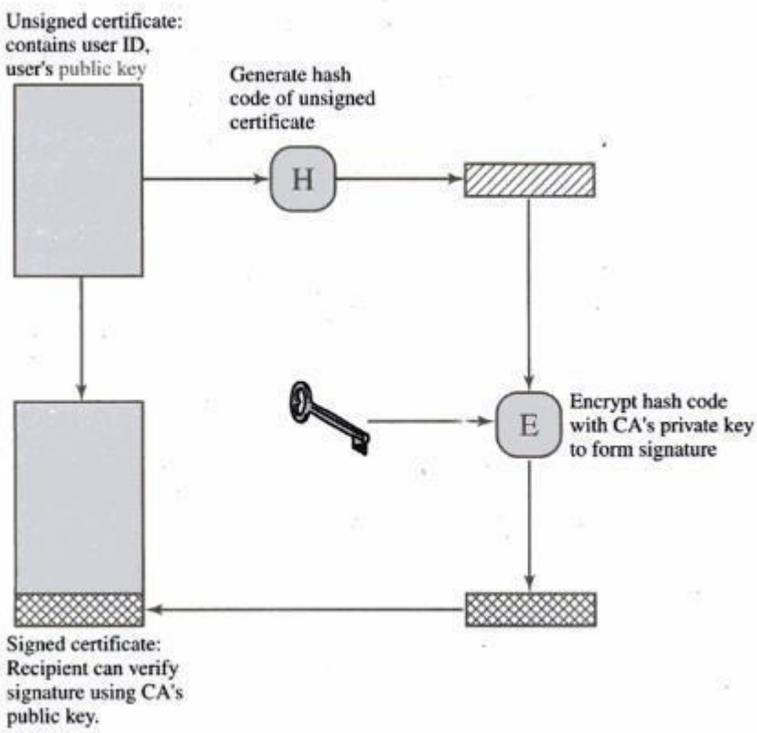


Figure 3.12 Public-Key Certificate Use

applications, including IP security, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME, all of which are discussed in Part Two. X.509 is examined in detail in Chapter 4.

Public-Key Distribution of Secret Keys

With conventional encryption, a fundamental requirement for two parties to communicate securely is that they share a secret key. Suppose Bob wants to create a messaging application that will enable him to exchange e-mail securely with anyone who has access to the Internet or to some other network that the two of them share. Suppose Bob wants to do this using conventional encryption. With conventional encryption, Bob and his correspondent, say, Alice, must come up with a way to share a unique secret key that no one else knows. How are they going to do that? If Alice is in the next room from Bob, Bob could generate a key and write it down on a piece of paper or store it on a diskette and hand it to Alice. But if Alice is on the other side of the continent or the world, what can Bob do? He could encrypt this key using conventional encryption and e-mail it to Alice, but this means that Bob and Alice must share a secret key to encrypt this new secret key. Furthermore, Bob and everyone else who uses this new e-mail package faces the same problem with every potential correspondent: Each pair of correspondents must share a unique secret key.

One approach is the use of Diffie-Hellman key exchange. This approach is indeed widely used. However, it suffers the drawback that, in its simplest form, Diffie-Hellman provides no authentication of the two communicating partners.

A powerful alternative is the use of public-key certificates. When Bob wishes to communicate with Alice, Bob can do the following:

1. Prepare a message
2. Encrypt that message using conventional encryption with a one-time conventional session key.
3. Encrypt the session key using public-key encryption with Alice's public key.
4. Attach the encrypted session key to the message and send it to Alice.

Only Alice is capable of decrypting the session key and therefore of recovering the original message. If Bob obtained Alice's public key by means of Alice's public-key certificate, then Bob is assured that it is a valid key.

CRYPTOGRAPHY AND NETWORK SECURITY
UNIT-IV

USER AUTHENTICATION

REMOTE USER AUTHENTICATION PRINCIPLES:

The process of verifying an identity claimed by or for a system entity. An authentication process consists of two steps:

- *Identification step:* Presenting an identifier to the security system. (Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.)
- *Verification step*: Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

There are four general means of authenticating a user's identity, which can be used alone or incombination:

- **Something the individual knows:** Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions.
- **Something the individual possesses:** Examples include cryptographic keys, electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a token.
- **Something the individual is (static biometrics):** Examples include recognition by fingerprint, retina, and face.
- **Something the individual does (dynamic biometrics):** Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

For network-based user authentication, the most important methods involve cryptographic keys and something the individual knows, such as a password.

Principle 1: Mutual Authentication:

Mutual authentication protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

Principle 2: One-Way Authentication:

Typically, the recipient wants some assurance that the message is from the alleged sender.

REMOTE USER AUTHENTICATION USING SYMMETRIC ENCRYPTION:

Mutual Authentication

Protocol of Needham and Schroeder for secret key distribution using a KDC can be summarized as follows.

Protocol-1

1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. A → B: $E(K_b, [K_s \| ID_A])$
4. B → A: $E(K_s, N_2)$
5. A → B: $E(K_s, f(N_2))$

- Secret keys Ka and Kb are shared between A and the KDC and B and the KDC, respectively.
- The purpose of the protocol is to distribute securely a session key Ks to A and B. A securely acquires a new session key in step 2.
- The message in step 3 can be decrypted, and hence understood, only by B.
- Step 4 reflects B's knowledge of Ks, and step 5 assures B of A's knowledge of ks and assures B that this is a fresh message because of the use of the nonce N2 .

Steps 4 and 5, the protocol is still vulnerable to a form of replay attack. Improved version onprotocol-1 is as follows:

Protocol-2

Protocol 2 Includes the addition of a timestamp to steps 2 and 3. T is a timestamp that assures Aand B that the session key has only just been generated.

1. A → KDC: $ID_A \| ID_B$
2. KDC → A: $E(K_a, [K_s \| ID_B \| T \| E(K_b, [K_s \| ID_A \| T])])$
3. A → B: $E(K_b, [K_s \| ID_A \| T])$
4. B → A: $E(K_s, N_1)$
5. A → B: $E(K_s, f(N_1))$

One-Way Authentication

Using symmetric encryption, the decentralized key distribution scenario illustrated for a message with content, the sequence is as follows:

1. A → KDC: $ID_A \| ID_B \| N_1$
2. KDC → A: $E(K_a, [K_s \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$
3. A → B: $E(K_b, [K_s \| ID_A]) \| E(K_s, M)$

KERBEROS:

Kerberos is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this:

Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services.

In particular, the following three threats exist:

- A user may gain access to a particular workstation and pretend to be another user operating

from that workstation.

- A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
- A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

Requirements of Kerberos

Kerberos listed the following requirements.

- *Secure*: A network eavesdropper should not be able to obtain the necessary information to impersonate a user.
- *Reliable*: Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.
- *Transparent*: Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.
- *Scalable*: The system should be capable of supporting large numbers of clients and servers.

Kerberos realm

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers collectively called as Kerberos realm which requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.

2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

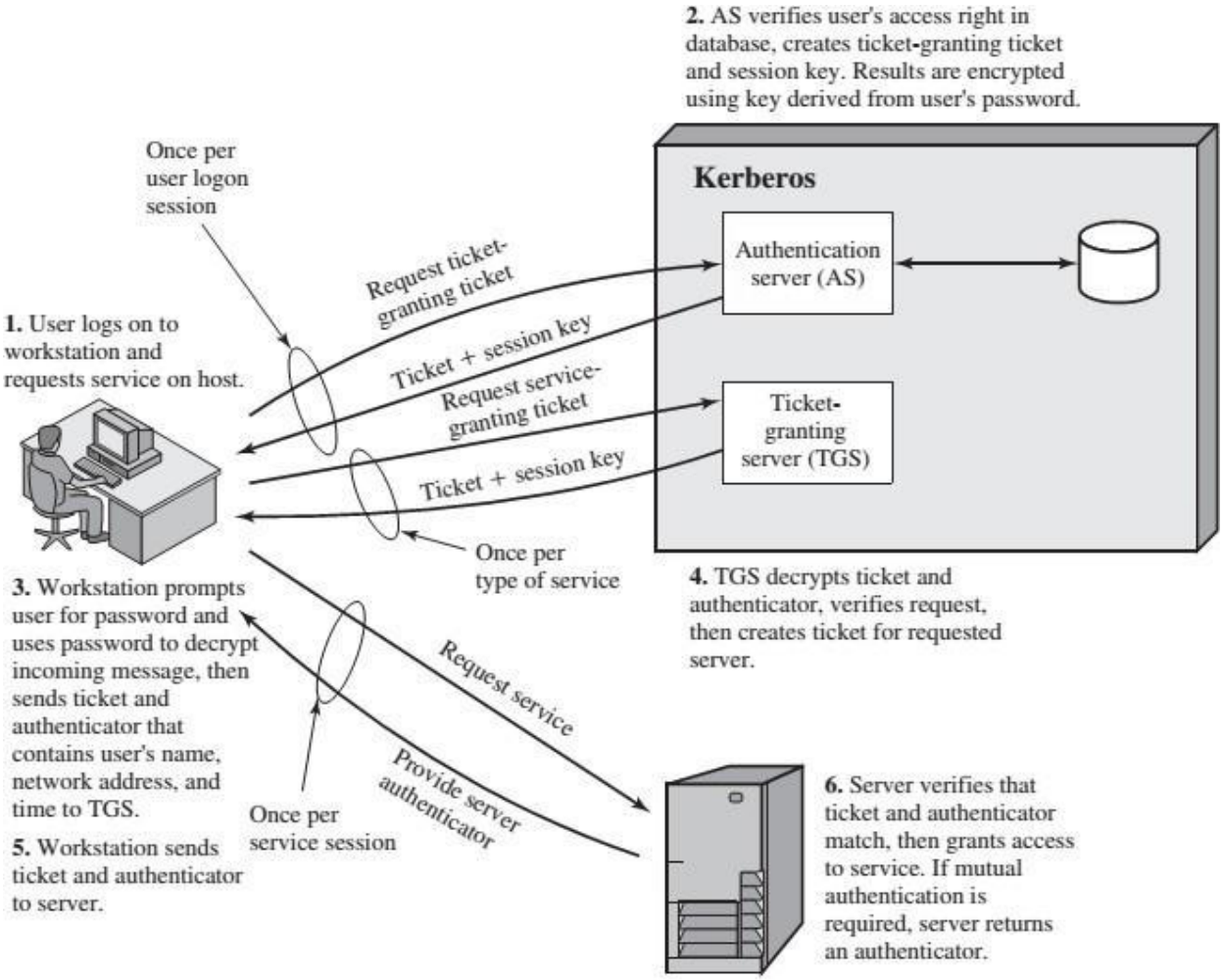


Figure 15.1 Overview of Kerberos

Details of Kerberos working

Once per user logon session:

(1) $C \rightarrow AS$: $ID_C \| ID_{tgs}$

(2) $AS \rightarrow C$: $E(K_c, Ticket_{tgs})$

Once per type of service:

(3) $C \rightarrow TGS$: $ID_C \| ID_V \| Ticket_{tgs}$

(4) $TGS \rightarrow C$: $Ticket_v$

Once per service session:

(5) $C \rightarrow V$: $ID_C \| Ticket_v$

$$Ticket_{tgs} = E(K_{tgs}, [ID_C \| AD_C \| ID_{tgs} \| TS_1 \| Lifetime_1])$$
$$Ticket_v = E(K_v, [ID_C \| AD_C \| ID_v \| TS_2 \| Lifetime_2])$$

Notations used are as follows:

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user requests access to TGS.
TS_1	Allows AS to verify that client's clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2).
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

Communication between two Realm

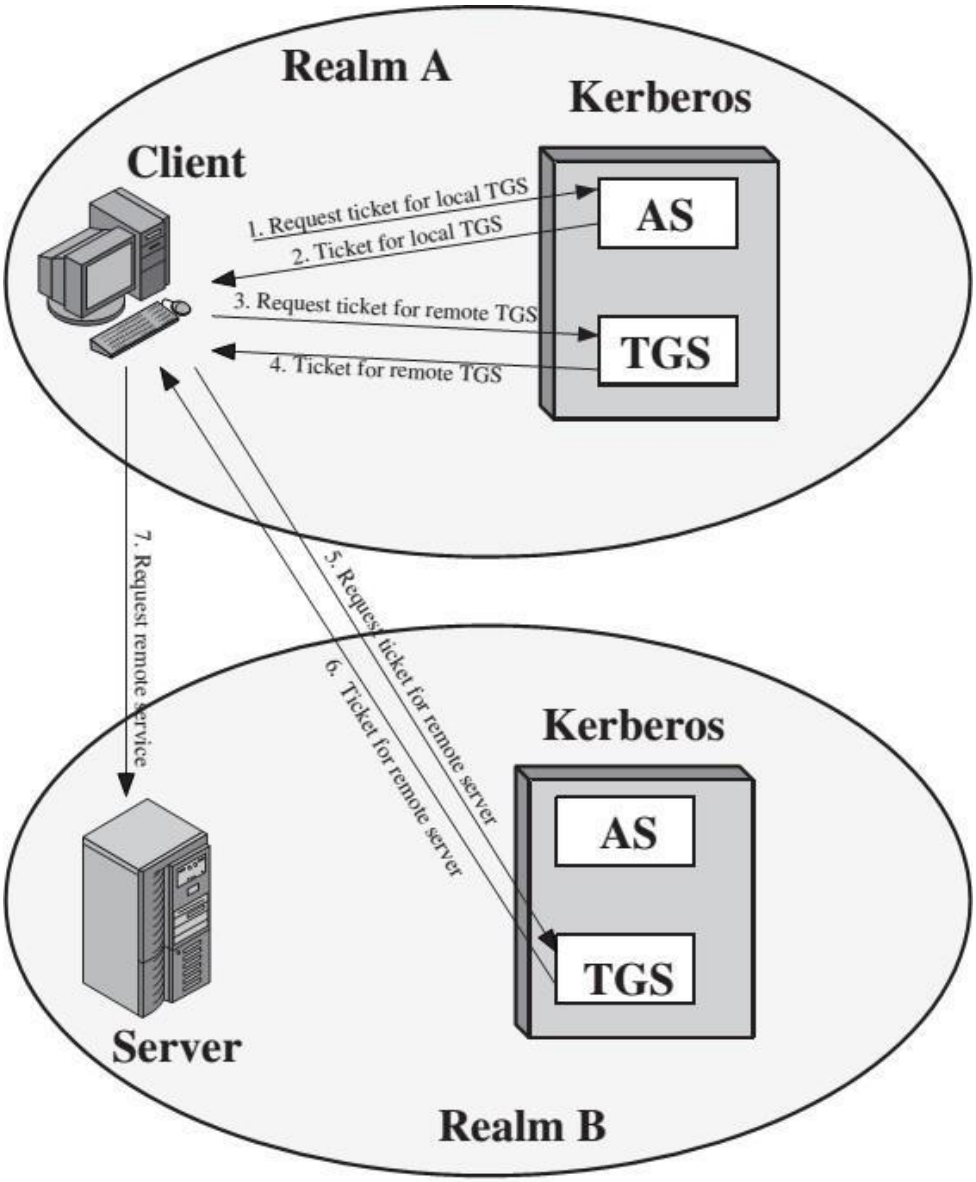


Figure 15.2 Request for Service in Another Realm

Electronic Mail Security:

Email is one of the most widely used and regarded network services. Currently message contents are not secure, may be inspected either in transit or by suitably privileged users on destination system.

Email Security Enhancements:

- confidentiality
    - protection from disclosure
- authentication
    - of sender of message
- message integrity
    - protection from modification
- non-repudiation of origin
    - protection from denial by sender

Pretty Good Privacy (PGP):

- Provides a confidentiality and authentication service that can be used for electronic mail and file storage applications
- Developed by Phil Zimmermann
    - Selected the best available cryptographic algorithms as building blocks
    - Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands
    - Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks
    - Entered into an agreement with a company to provide a fully compatible, low-cost commercial version of PGP

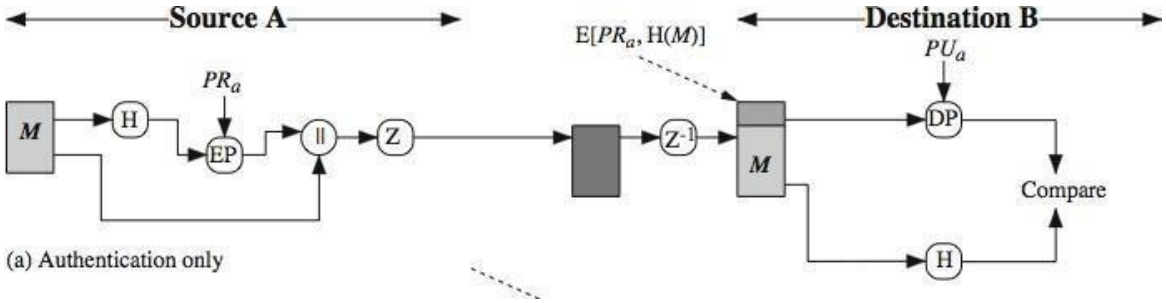
PGP Growth:

- It is available free worldwide in versions that run on a variety of platforms
- The commercial version satisfies users who want a product that comes with vendor support
- It is based on algorithms that have survived extensive public review and are considered extremely secure
- It has a wide range of applicability
- It was not developed by, nor is it controlled by, any governmental or standards organization
- Is now on an Internet standards track, however it still has an aura of an antiestablishment endeavor

PGP Operation – Authentication:

1. sender creates a message
2. SHA-1 used to generate 160-bit hash code of message
3. hash code is encrypted with RSA using the sender's private key, and result is attached to message
4. receiver uses RSA or DSS with sender's public key to decrypt and recover hash code

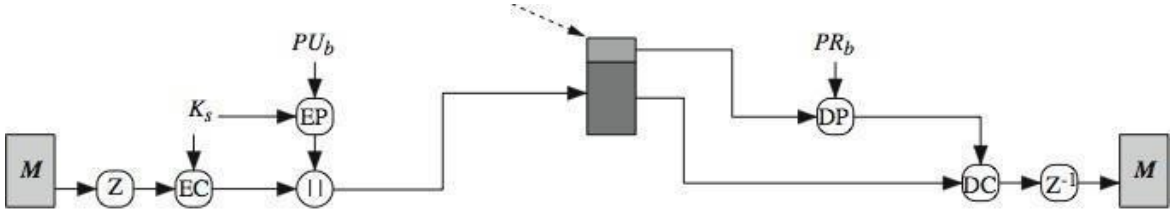
5. receiver generates new hash code for message and compares with decrypted hash code, if match, message is accepted as authentic



(a) Authentication only

PGP Operation – Confidentiality:

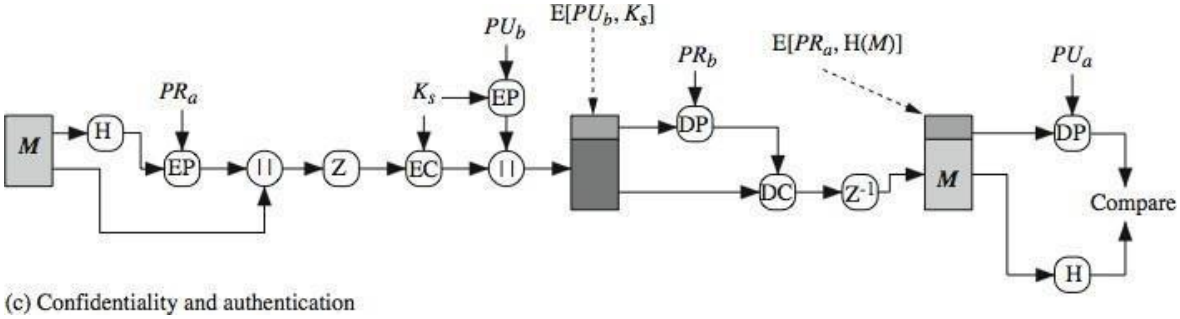
1. sender generates message and random 128-bit number to be used as session key for this message only
2. message is encrypted, using CAST-128 / IDEA/3DES with session key
3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message



(b) Confidentiality only

PGP Operation – Confidentiality & Authentication:

- uses both services on same message
  - create signature & attach to message
  - encrypt both message & signature
  - attach RSA encrypted session key



(c) Confidentiality and authentication

PGP Operation – Compression:

- by default, PGP compresses message after signing but before encrypting
    - so can store uncompressed message & signature for later verification
    - & because compression is non deterministic
- uses ZIP compression algorithm

PGP Operation – Email Compatibility:

- when using PGP will have binary data to send (encrypted message etc)
- however, email was designed only for text
- hence PGP must encode raw binary data into printable ASCII characters
- uses radix-64 algorithm
    - maps 3 bytes to 4 printable chars
    - also appends a CRC
- PGP also segments messages if too big

S/MIME (Secure/Multipurpose Internet Mail Extensions):

- It is a security enhancement to MIME Internet e-mail format standard based on technology from RSA Data Security
- Defined in:
    - RFCs 3370, 3850, 3851, 3852
- S/MIME support in many mail agents
    - eg MS Outlook, Mozilla, Mac Mail etc
- To understand S/MIME, we need first to have a general understanding of the underlying e-mail format that it uses, namely MIME. We have to learn about RFC5322(internet Message Format)

RFC 5322:

- Defines a format for text messages that are sent using electronic mail
- Messages are viewed as having an envelope and contents
    - The envelope contains whatever information is needed to accomplish transmission and delivery
    - The contents compose the object to be delivered to the recipient
    - RFC 5322 standard applies only to the contents
- The content standard includes a set of header fieldsthat may be used by the mail system to create the envelope

Multipurpose Internet Mail Extensions (MIME):

- An extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP)
- Is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations
- The specification is provided in RFCs 2045 through 2049

The MIME specification includes the following elements.

1. **Five new message header fields** are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimediaelectronicmail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

The Five Header Fields Defined in MIME:

The five header fields defined in MIME are

- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to representthe body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

S/MIME Functionality:

S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encrypted content encryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signatureare then encoded using base64 encoding. A signed data message can only be viewed by a recipientwith S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However,in this case, only the digital signature is encoded using base64. As a result, recipients withoutS/MIME capability can view the message content, although they cannot verify the signature.
- **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

S/MIME Messages:

- S/MIME secures a MIME entity with a signature, encryption, or both.
- forming a MIME wrapped **Public-Key Cryptography Standards(**PKCS) object
- have a range of content-types:
    - enveloped data
    - signed data
    - clear-signed data
    - registration request
    - certificate only message

S/MIME Certificate Processing:

- S/MIME uses public-key certificates that conform to version 3 of X.509
- The key-management scheme used by S/MIME is in some ways a hybrid betweena strict X.509 certification hierarchy and PGP's web of trust
- S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists
    - The responsibility is local for maintaining the certificates needed to verify incoming signatures and to encrypt outgoing messages
- The certificates are signed by certification authorities

IP SECURITY OVERVIEW:

IPSec is an Internet Engineering Task Force (IETF) standard suite of protocols that provides data authentication, integrity, and confidentiality as data is transferred between communication points across IP networks. IPSec provides data security at the IP packet level. A packet is a data bundle that is organized for transmission across a network, and it includes a header and payload (the data in the packet). IPSec emerged as a viable network security standard because enterprises wanted to ensure that data could be securely transmitted over the Internet. IPSec protects against possible security exposures by protecting data whilein transit.

IPSEC SECURITY FEATURES:

IPSec is the most secure method commercially available for connecting network sites. IPSec wasdesigned to provide the following security features when transferring packets across networks:

- Authentication: Verifies that the packet received is actually from the claimed sender.
- Integrity: Ensures that the contents of the packet did not change in transit.
- Confidentiality: Conceals the message content through encryption.

IPSEC ELEMENTS:

IPSec contains the following elements:

- **Encapsulating Security Payload (ESP):** Provides confidentiality, authentication, and integrity.
- **Authentication Header (AH):** Provides authentication and integrity.
- **Internet Key Exchange (IKE):** Provides key management and Security Association (SA) management.

APPLICATIONS OF IPSEC:

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following:

► Secure branch office connectivity over the Internet

► Secure remote access over the Internet

► **Establishing extranet and intranet connectivity with partners**: IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing akey exchange mechanism.

► **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

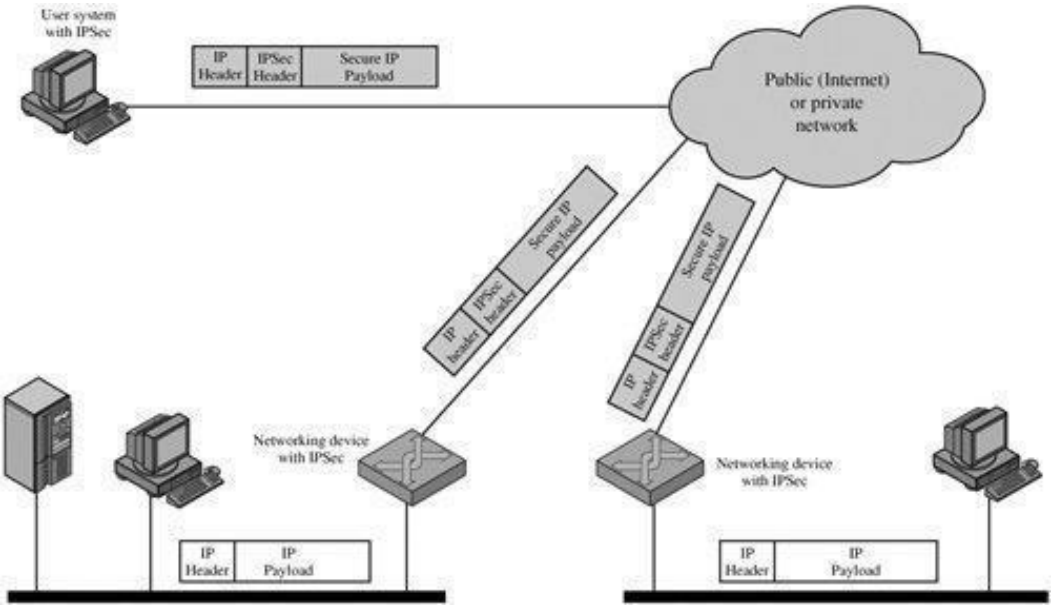


Figure. An IP Security Scenario

BENEFITS OF IPSEC:

- IPSec provides strong security within and across the LANs.
- Firewall uses IPSec to restrict all those incoming packets which are not using IP. Since
- firewall is the only way to enter into an organization, restricted packets cannot enter.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications.
- There is no need to change software on a user or server system when IPSec is implemented in the firewall or router. Even if IPSec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPSec can be transparent to end users.
- IPSec can provide security for individual users if needed.

IP SECURITY ARCHITECTURE:

Mainly the IPSec is constituted by three major components.

- ► IPSec Documents
- ► IPSec Services
- ► Security Associations(SA)

IPSec Documents:

The IPSec specification consists of numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408:

- ► RFC 2401: An overview of a security architecture
- ► RFC 2402: Description of a packet authentication extension to IPv4 and IPv6
- ► RFC 2406: Description of a packet encryption extension to IPv4 and IPv6
- ► RFC 2408: Specification of key management capabilities

The header for authentication is known as the Authentication header(AH); that for encryption is known as the **Encapsulating Security Payload (ESP)** header.

The documents are divided into seven groups, as depicted in Figure

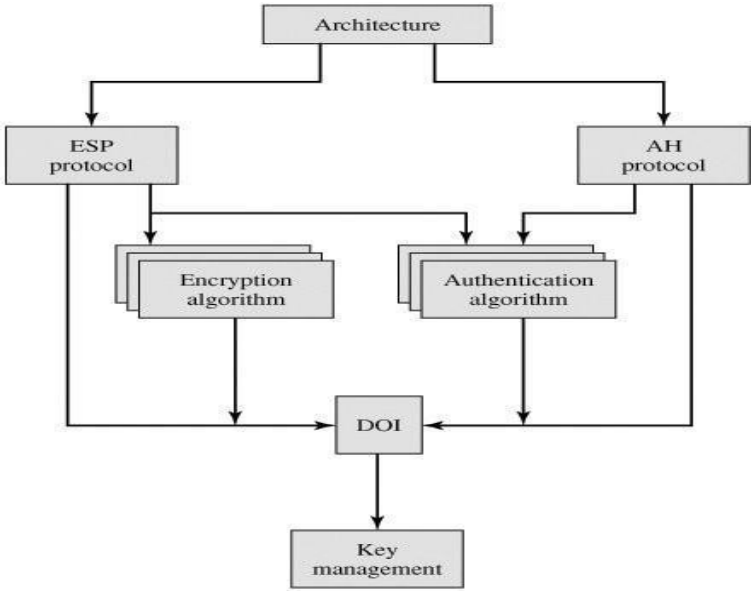


Figure. IPSec Document Overview

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPSectechnology
- .**Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.
- **Authentication Header (AH):** Covers the packet format and general issues related to theuse of AH for packet authentication.
- **Encryption Algorithm:** A set of documents that describe how various encryption algorithms are used for ESP.
- **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.
- **Key Management:** Documents that describe key management schemes.
- **Domain of Interpretation (DOI):** Contains values needed for the other documents to relate to each other. These include identifiers for approvedencryptionand authenticationalgorithms, as well as operational parameters such as key lifetime.

IPSec Services:

IPSec provides security services at the IP layer by selecting requiredsecurityprotocols, algorithms and cryptographickeys as per the services requested.

Two protocols are used to provide security:

- an authentication protocol designated by the header of the protocol, **Authentication Header (AH)**
- and a combined encryption/authentication protocol designated by the format of the packet for that protocol, **Encapsulating Security Payload (ESP).**

The services are

- ► Access control
- ► Connectionless integrity
- ► Data origin authentication
- ► Rejection of replayed packets
- ► Confidentiality
- ► Limited traffic flow confidentiality

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

Security Associations:

A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but notboth.

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. SPI is located in AH and ESP headers. SPI enables the receiving system under which the packet is to process.
- **IP Destination Address:** It is the end point address of SA which can be end user system or a network system.
- **Security Protocol Identifier:** security protocol identifier indicates whether the associationsis an AH or ESP.

SA Parameters:

The implementation of IPSec contain SA database which identifies the parameters related to SA.

- **Sequence Number Counter**: A 32-bit value usedto generate the Sequence Number field in AHor 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.
- **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.
- **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 or terminated.

- **IPSec Protocol Mode:** This parameter represents the type of mode used for IPSec implementation. The mode may be a Tunnel or transport.

- **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet thatcan be transmitted).

SA Selectors:

- IPSec provides flexibility in providing services to the users according to their needs. For this purpose, SA's are used. Different combinations of SA's can give different user configurations.IPSec is also capable of differentiating traffic i.e., which traffic is allowed to pass and which traffic should be forwarded the IPSec protection. The property of IPSec requires the traffic tobe associated with a security association. To associate a particular SA to IP traffic IPSec maintains a database called Security Policy Database(SPD).
- SPD is table entries which maps a set of IP traffic to a single or more SAs.
- Selectors are basically used to define policy that specifies which packet should be forwardedand which packet should be rejected to filter outgoing traffic.

A sequence of steps is performed on the outgoing traffic,

- Compare the values of the appropriate fields in the packet (the selector fields) against the SPDto find a matching SPD entry, which will point to zero or more SAs.
- Determine the SA if any for this packet and its associated SPI. **Security Parameter Index(SPI**) is one of the fields of IPSec header which is a unique identifier to identify a security association.
- Do the required IPSec processing (i.e., AH or ESP processing).The

following selectors determine an SPD entry:

- **Destination IP Address:** This may be a single IP address, an enumerated list or range of addresses.
- **Source IP Address:** This may be a single IP address, an enumerated list or range of addresses.
- **UserID:** A user identifier from the operating system. This is not a field in the IP or upper-layer headers but is available if IPSec is running on the same operating system as the user.
- **Data Sensitivity Level:** Used for systems providing information flow security (e.g., Secret or Unclassified).
- **Transport Layer Protocol:** Obtained from the IPv4 Protocol or IPv6 Next Header field. This may be an individual protocol number, a list of protocol numbers, or a range of protocol numbers.
- **Source and Destination Ports:** These may be individual TCP or UDP port values, an enumerated list of ports, or a wildcard port.

Transport and Tunnel Modes:

- Both AH and ESP support two modes of use: **transport and tunnel mode**.
- The operation of these two modes is best understood in the context of a descri ption of AH and ESP.

Transport Mode:

Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet. Transport mode is used for end-to-end communication between two hosts. ESP in transport mode encrypts and optionallyauthenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.

Tunnel Mode:

Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header. The entire original, or inner, packet travels through a "tunnel" from one point

of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.

Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewallor router that implements IPSec. ESP in tunnel mode encrypts and optionally authenticates the entireinner IP packet, including the inner IP header. AH in tunnel mode authenticates the entire inner IP packet and selected portions of the outer IP header.

AUTHENTICATION HEADER(AH):

The Authentication Header provides support for data integrity and authentication of IP packets. Dataintegrity service insures that data inside IP packets is not altered during the transit. The authentication feature enables an end system to authenticate the user or application and filter traffic accordingly. It also prevents the **address spoofing attacks** (A technique used to gain unauthorized access to computers, whereby the intruder sends messages to a computer with an IP **address** indicating that the message is coming from a trusted host).Authentication is based on the use of a message authentication code (MAC)i.e.; two communication parties must share a secret key.

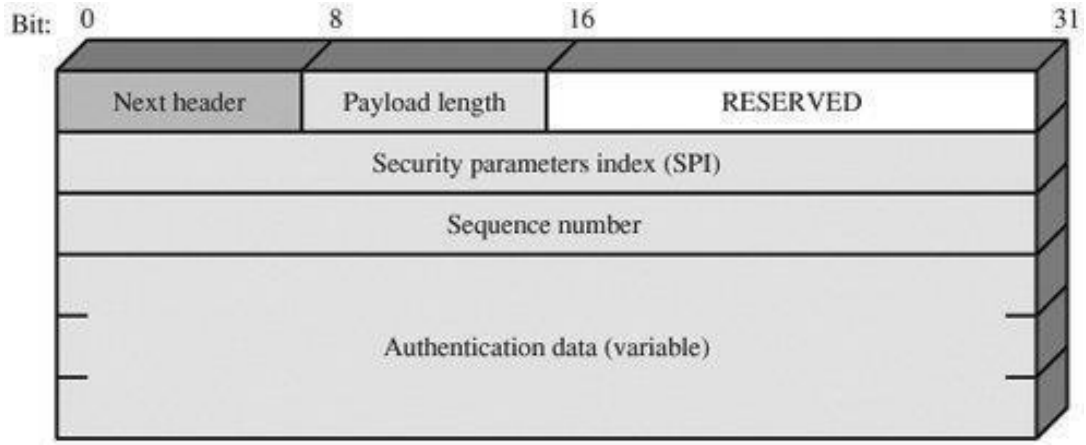


Figure. IPSec Authentication Header

The Authentication Header consists of the following fields

1. **Next Header (8 bits):** Identifies the type of header that immediately following the AH.
2. **Payload Length:** Length of Authentication Header in 32-bit words.
3. **Reserved (16 bits):** For future use.
4. **Security Parameters Index (32 bits):** Identifies a security association.
5. **Sequence Number (32 bits):** A monotonically increasing counter value.
6. **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet.

Anti-Replay Service:

A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits it to the intendeddestination. The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence. The **Sequence Number** fieldis designed to stop such attacks.

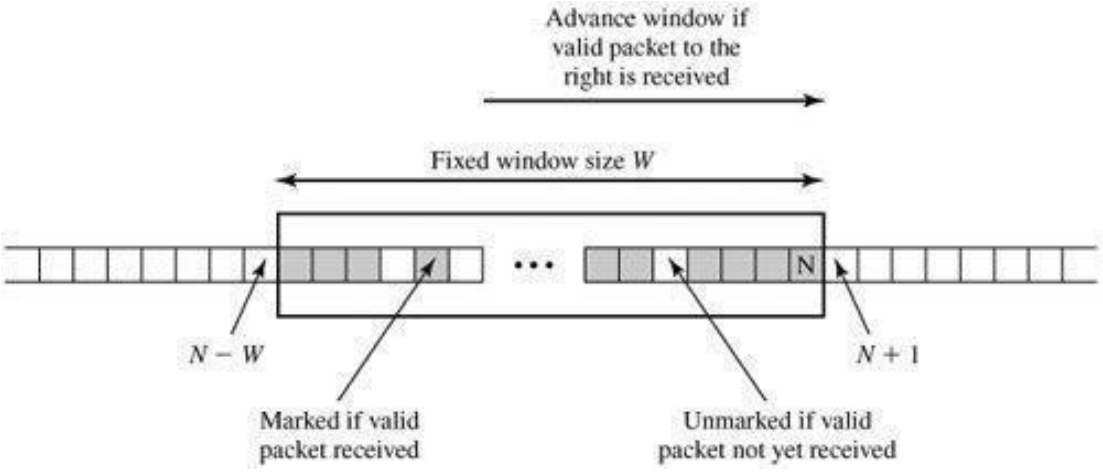


Figure. Antireplay Mechanism

When a new SA is established, the senderinitializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. If anti-replay is enabled (the default), the sender

must not allow the sequence number to cycle past 2^{32}-1 back to zero. Otherwise, there would be

multiple valid packets with the same sequence number. If the limit of 2^{32}-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key. IP is a connectionless, unreliable service, the protocol does not guarantee that packets will be delivered in order and does not guarantee that all packets will be delivered. Therefore, the IPSec authentication document dictates that the receiver should implement a window of size W, with a default of W = 64. The right edge of the window represents the highest sequence number, N, so far received for a valid packet. For any packet with a sequence number in the range from N-W+ 1 to N that has been correctly received (i.e., properly authenticated), the corresponding slot in the windowis marked (Figure).

Inbound processing proceeds as follows when a packet is received:

1. If the received packet falls within the windowand is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.
2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.
3. If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

Integrity Check Value:

The Authentication Data field holds a value referred to as the Integrity Check Value. The ICV is amessage authentication code or a truncated version of a code produced by a MAC algorithm.

Transport and Tunnel Modes:

There are two ways in which the IPSec authenticationservice can be used. In one case, **authenticationis provided directly** between a server and client workstations; the workstation can be either on the same network as the server or on an external network. As long as the workstation and the server sharea protected secret key, the authentication process is secure. This case uses a **transport mode SA**. In the other case, a **remote workstation authenticates itself to the corporate firewall**, either for accessto the entire internal network or because the requested server does not support the authentication feature. This case uses a **tunnel mode SA**.

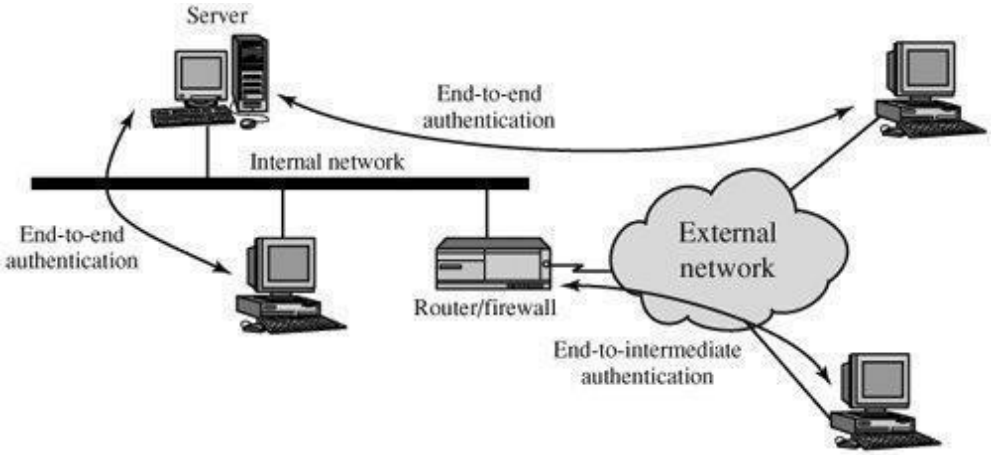


Figure. End-to-End versus End-to-Intermediate Authentication

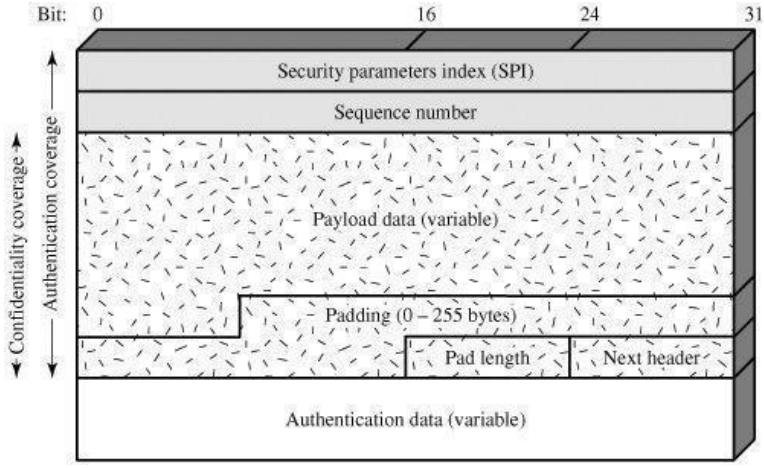
Encapsulating Security Payload(ESP):

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format:

It contains the following fields:

1. **Security Parameters Index (32 bits):** Identifies a security association.
2. **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
3. **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
4. **Padding (0-255 bytes):** The purpose of this field is discussed later.
5. **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
6. **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that.
7. **Authentication Data (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.



Encryption and Authentication Algorithms:

The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESP.

Various algorithms used for encryption are: Three-key triple DES, RC5, IDEA, Three-key triple IDEA,CAST, Blowfish

Padding:

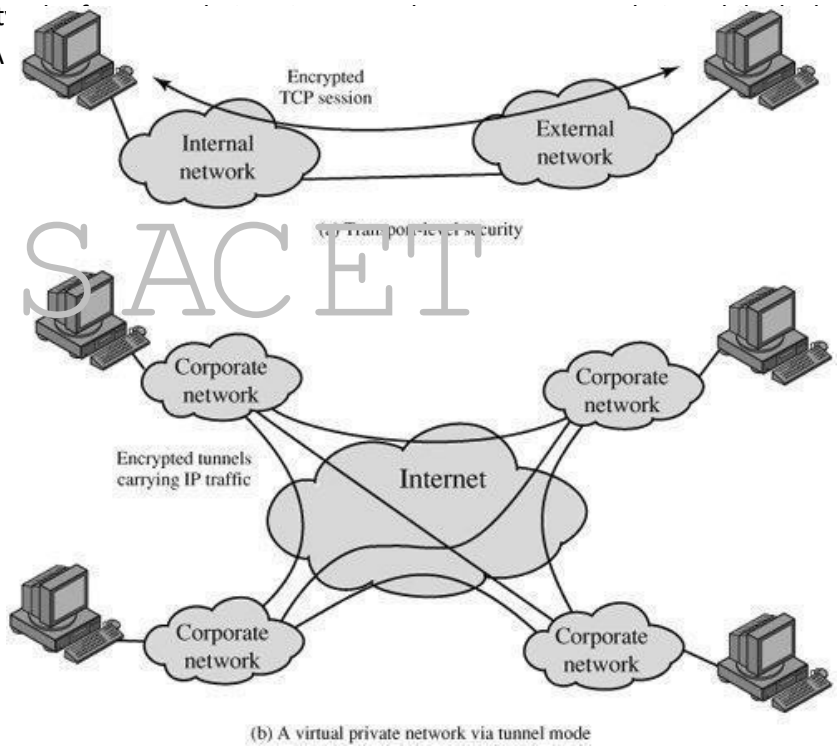
The Padding field serves several purposes:

1. If an encryption algorithm requires the plaintext to be a multiple of some number of bytes. The Padding field is used to expand the plaintext to the required length.
2. The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.
3. Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

Transport and Tunnel Modes:

Figure shows two ways in which the IPSec ESP service can be used. In the upper part of the figure,

encryption (and optionally authentication) is provided directly between two hosts. Figure(b) shows how tunnel mode operation can be used to set up a *virtual private network*. In this example, an organization has four private networks interconnected across the Internet. Hosts on the internal networks use the Internet for transport of data but do not interact with other Internet-based hosts. By terminating the tunnels at the security gateway to each internal network, the configuration allowsthe hosts to avoid implementing the security capabilit tter technique uses atunnel mode SA



(b) A virtual private network via tunnel mode

COMBINING SECURITY ASSOCIATIONS:

An individual SA can implement either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for the services providedby both AH and ESP. Further, a particular traffic flow mayrequire IPSec

services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPSec services. The term ***security association bundle*** refers to a sequenceof SAs through which trafficmust be processed to provide a desired set of IPSec services.

Security associations may be combined into bundles in two ways:

- **Transport adjacency:** Refers to applying more than one security protocol to the same IPpacket, withoutinvoking tunneling.
- **Iterated tunneling:** Refers to the application of multiple layers of security protocols effectedthrough IP tunneling.

KEY MANAGEMENT:

The key management portion of IPSec involves the determination and distribution of secret keys. A typical requirement is four keys for communication between two applications: transmit and receive pairs for both AH and ESP.

The IPSec Architecture document mandates support for two types of key management:

- **Manual:** A system administrator manually configures each system with its own keys and withthe keys of other communicating systems. This is suitable for small, relatively static environments.
- **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system.

The default automated key management protocol for IPSec is referred to as ISAKMP/Oakley andconsists of the following elements:

1. **Oakley Key Determination Protocol**
2. **Internet Security Association and Key Management Protocol (ISAKMP)**

Oakley Key Determination Protocol:

Oakley is a key exchange protocol basedon the Diffie-Hellman algorithmbut providingadded security.Oakley is genericin that it does notdictate specific formats.

The Diffie-Hellman algorithm has **two** attractive features:

1. Secret keys are created only when needed.
2. The exchange requires no preexisting infrastructure other than an agreement on the global parameters.

However, there are a number of weaknesses to Diffie-Hellman, as pointed out in

3. It does not provide any information about the identities of the parties.
4. It is subject to a man-in-the-middle attack

It is computationally intensive. As a result, it is vulnerable to a clogging attack, in which an opponent requests ahigh number of keys.

Oakley is designed to retain the advantages of Diffie-Hellman while countering its weaknesses.

Features of Oakley:

The Oakley algorithm is characterized by five important features:

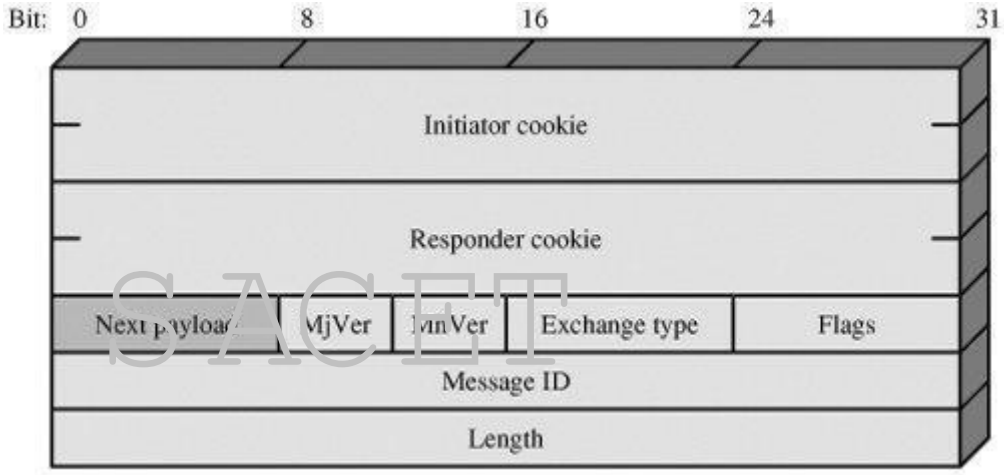
- ► It employs a mechanism known as cookies to thwart clogging attacks.
  - ► It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.
- ► It uses nonces to ensure against replay attacks.
- ► It enables the exchange of Diffie-Hellman publickey values.
- ► It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

Internet Security Association and Key Management Protocol (ISAKMP):

ISAKMP provides a framework for Internet key management and provides the specific protocolsupport, including formats, for negotiation of security attributes.

ISAKMP Header Format:

An ISAKMP message consists of an ISAKMP header followed by one or more payloads. All of this is carried in a transport protocol. The specification dictates that implementations must support the useof UDP for the transport protocol.



(a) ISAKMP header

It consists of the following fields:

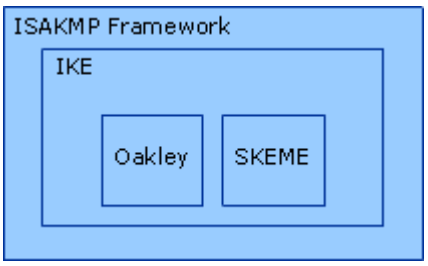
1. **Initiator Cookie (64 bits):** Cookie of entity that initiated SA establishment, SA notification, orSA deletion.
2. **Responder Cookie (64 bits):** Cookie of responding entity; null in first message from initiator.
3. **Next Payload (8 bits):** Indicates the type of the first payload in the message
4. **Major Version (4 bits):** Indicates major version of ISAKMP in use.
5. **Minor Version (4 bits):** Indicates minor version in use.
6. **Exchange Type (8 bits):** Indicates the type of exchange.
7. **Flags (8 bits):** Indicates specific options set for this ISAKMP exchange.
8. **Message ID (32 bits):** Unique ID for this message.
9. **Length (32 bits):** Length of total message (header plus all payloads) in octets.

INTERNET KEY EXCHANGE:

Internet Key Exchange (IKE) is a key management protocol standard used in conjunction

with the Internet Protocol Security (IPSec) standard protocol. It can also be described as a method for exchanging keys for encryption and authentication over an unsecured medium, such as the Internet.

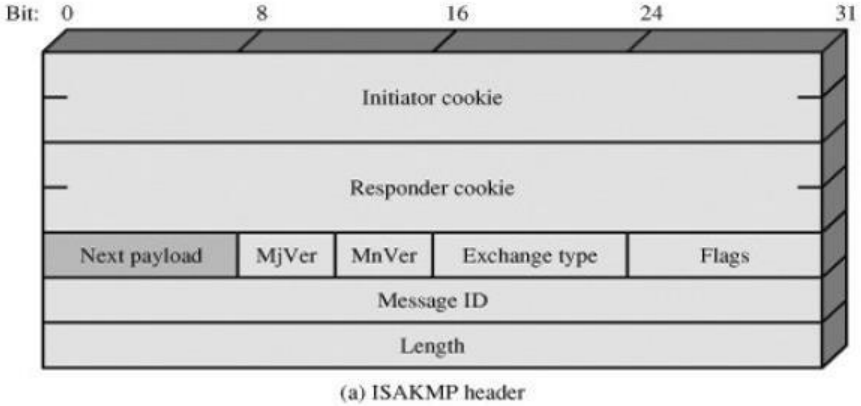
IKE is a hybrid protocol based on:



- ISAKMP
  - **Oakley**:
- SKEME
  -

1. **ISAKMP** : Internet Security Association and Key Management Protocols(ISAKMP) are used for negotiation and establishment of security associations. It's not a key exchange protocol , it's a framework on which key exchange protocols operate.

   ISAKMP Header Format
   - **Initiator Cookie (64 bits):** Cookie of entity that initiated SA establishment, SA notification, or SA deletion.
   - **Responder Cookie (64 bits):** Cookie of responding entity; null in first message from initiator.
   - **Next Payload (8 bits):** Indicates the type of the first payload in the message
   - **Major Version (4 bits):** Indicates major version of ISAKMP in use.
   - **Minor Version (4 bits):** Indicates minor version in use.
   - **Exchange Type (8 bits):** Indicates the type of exchange;
   - **Flags (8 bits):** Indicates specific options set for this ISAKMP exchange.
   - **Message ID (32 bits):** Unique ID for this message.
   - **Length (32 bits):** Length of total message



(a) ISAKMP header

2. **Oakley**: This protocol is used for key agreement or key exchange. Oakley defines the mechanism that is used for key exchange over an IKE session. The default algorithm forkey exchange used by this protocol is the Diffie-Hellman algorithm.

Features of Oakley

The Oakley algorithm is characterized by five important features:

1. It employs a mechanism known as cookies to thwart clogging attacks.
2. It enables the two parties to negotiate a group; this, in essence, specifies the globalparameters of the Diffie-Hellman key exchange.
3. It uses nonces to ensure against replay attacks.
4. It enables the exchange of Diffie-Hellman public key values.
5. It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

Consider the problem of **clogging attacks**. In this attack, an opponent **forges** the source address of a legitimate user and sends a public Diffie-Hellman key to the victim. The victim then performs a modular exponentiation to compute the secret key. Repeated messages of this type can clog the victim's system with useless work. The **cookie exchange** requires that each side send a pseudorandom number, the cookie, in the initial message, which the other side acknowledges. This acknowledgment must be repeated in the first message of the Diffie-Hellman key exchange. If the source address was forged, the opponent gets no answer. Thus, an opponent can only force a user to generate acknowledgments and not to perform the Diffie-Hellman calculation.

- **SKEME**: This protocol is another version for key exchange. Provides support for public- key-based key exchange, key distribution centres, and manual installation, it also outlines methods of secure and fast key refreshment. A secure and versatile key exchange protocol for key management over Internet is presented. SKEME constitutes a compact protocolthat supports a variety of realistic scenarios and security models over Internet.

Unit-5
TRANSPORT LEVEL SECURITY

WEB SECURITY REQUIREMENTS:

Transport-level security, often referred to as Transport Layer Security (TLS) or its predecessor Secure Sockets Layer (SSL), is crucial for ensuring the security of data during its transmission over the Internet. Here are some key web security requirements in transport-level security:

1. **Encryption:**
    - **SSL/TLS Protocols:** Use the latest versions of SSL or TLS protocols to encrypt data during transmission. It's essential to stay updated with the latest security standards, as older versions may have vulnerabilities.
    - **Strong Encryption Algorithms:** Utilize strong cryptographic algorithms for encryption, such as Advanced Encryption Standard (AES). Avoid using weak algorithms that are susceptible to attacks.
2. **Certificates and Public Key Infrastructure (PKI):**
    - **Digital Certificates:** Obtain and use digital certificates from reputable Certificate Authorities (CAs) to ensure the authenticity of your website. This helps users verify that they are connecting to the intended server and not a malicious one.
    - **Certificate Validity:** Regularly check and renew SSL/TLS certificates to ensure they are valid. Expired or compromised certificates can expose your website to security risks.
3. **Perfect Forward Secrecy (PFS):**
    - **PFS Support:** Implement Perfect Forward Secrecy to ensure that even if a private key is compromised, past communications remain secure. This is achieved by generating unique session keys for each session.
4. **Secure Cipher Suites:**
    - **Disable Weak Cipher Suites:** Disable outdated and weak cipher suites that may be susceptible to attacks. Use only strong and secure cipher suites to ensure the confidentiality and integrity of data.
5. **HSTS (HTTP Strict Transport Security):**
    - **HSTS Header:** Implement HSTS to ensure that web browsers always connect to your site using a secure connection (HTTPS). This helps prevent man-in-the-middle attacks that attempt to downgrade the connection to HTTP.
6. **Server Name Indication (SNI):**
    - **SNI Support:** If hosting multiple websites on the same server, ensure that SNI is supported. SNI allows the server to present different SSL certificates based on the domain name requested by the client.

7. **Secure Configuration:**
    - **Disable Insecure Protocols:** Disable deprecated and insecure protocols like SSLv2 and SSLv3. Only support the latest and secure TLS versions.
    - **Secure Default Configurations:** Ensure that the server's default configurations are secure. Unnecessary services and features should be disabled or configured securely.
8. **Certificate Pinning:**
    - **Public Key Pinning:** Implement certificate pinning to bind a certificate to a specific public key. This helps prevent the acceptance of fraudulent certificates issued by compromised CAs.
9. **Monitoring and Logging:**
    - **Security Monitoring:** Implement continuous monitoring of security events related to SSL/TLS, including failed connection attempts, cipher suite negotiations, and certificate issues.
    - **Logging:** Keep detailed logs of SSL/TLS-related events for auditing and troubleshooting purposes.
10. **Regular Audits and Vulnerability Scans:**
    - **Regular Audits:** Conduct regular security audits to identify and address potential vulnerabilities in the SSL/TLS implementation.
    - **Vulnerability Scans:** Use automated tools to scan for vulnerabilities and weaknesses in your web server's SSL/TLS configuration.

By adhering to these web security requirements in transport-level security, you can help create a secure and trustworthy communication channel between your web server and clients. Keep in mind that security is an ongoing process, and regular updates and improvements are essential to stay ahead of emerging threats.

TRANSPORT LAYER SECURITY:

In order to provide an open ***Internet*** standard of SSL, IETF released The Transport Layer Security (TLS) protocol in January 1999. TLS is defined as a proposed Internet Standard in RFC 5246.

Salient Features

- TLS protocol has same objectives as SSL.
- It enables client/server applications to communicate in a secure manner by authenticating, preventing eavesdropping and resisting message modification.
- TLS protocol sits above the reliable connection-oriented transport TCP layer in the networking layers stack.
- The architecture of TLS protocol is similar to SSLv3 protocol. It has two sub protocols: the TLS Record protocol and the TLS Handshake protocol.
- Though SSLv3 and TLS protocol have similar architecture, several changes were made in architecture and functioning particularly for the handshake protocol.

Comparison of TLS and SSL Protocols

There are main eight differences between TLS and SSL protocols. These are as follows −

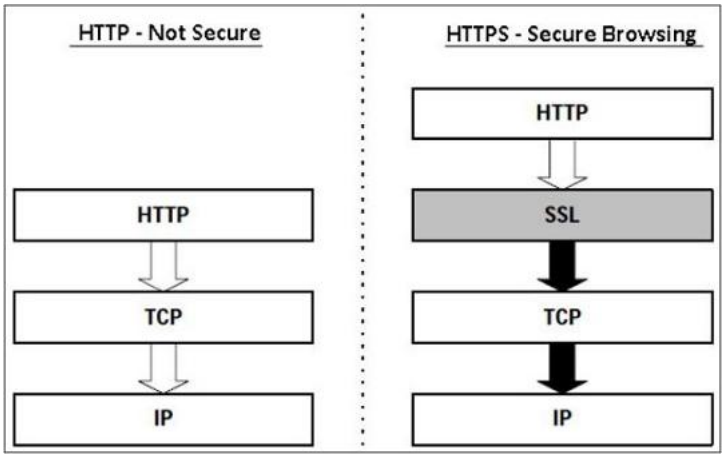
- **Protocol Version** − The header of TLS protocol segment carries the version number 3.1 to differentiate between number 3 carried by SSL protocol segment header.
- **Message Authentication** − TLS employs a keyed-hash message authentication code (H- MAC). Benefit is that H-MAC operates with any hash function, not just MD5 or SHA, as explicitly stated by the SSL protocol.
- **Session Key Generation** − There are two differences between TLS and SSL protocol for generation of key material.
    - Method of computing pre-master and master secrets is similar. But in TLS protocol, computation of master secret uses the HMAC standard and pseudorandom function (PRF) output instead of ad-hoc MAC.
    - The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.
- Alert Protocol Message −
    - TLS protocol supports all the messages used by the Alert protocol of SSL, except *No certificate* alert message being made redundant. The client sends empty certificate in case client authentication is not required.
    - Many additional Alert messages are included in TLS protocol for other error conditions such as *record_overflow, decode_error* etc.
- **Supported Cipher Suites** − SSL supports RSA, Diffie-Hellman and Fortezza cipher suites. TLS protocol supports all suits except Fortezza.
- **Client Certificate Types** − TLS defines certificate types to be requested in a *certificate_request* message. SSLv3 support all of these. Additionally, SSL support certain other types of certificate such as Fortezza.
- Certificate Verify and Finished Messages −
    - In SSL, complex message procedure is used for the *certificate_verify* message. With TLS, the verified information is contained in the handshake messages itself thus avoiding this complex procedure.
    - Finished message is computed in different manners in TLS and SSLv3.

The above differences between TLS and SSLv3 protocols are summarized in the following table

	SSL v3.0	TLS v1.0
Protocol version in messages	3.0	3.1
Alert protocol message types	12	23
Message authentication	ad hoc	standard
Key material generation	ad hoc	PRF
CertificateVerify	complex	simple
Finished	ad hoc	PRF
Baseline cipher suites	includes Fortezza	no Fortezza

HTTPS

Hyper Text Transfer Protocol (HTTP) protocol is used for web browsing. The function of HTTPS is similar to HTTP. The only difference is that HTTPS provides “secure” web browsing. HTTPS stands for HTTP over SSL. This protocol is used to provide the encrypted and authenticated connection between the client web browser and the website server.



The secure browsing through HTTPS ensures that the following content are encrypted −

- URL of the requested web page.
- Web page contents provided by the server to the user client.
- Contents of forms filled in by user.
- Cookies established in both directions.

Working of HTTPS

HTTPS application protocol typically uses one of two popular transport layer security protocols - SSL or TLS. The process of secure browsing is described in the following points.

- You request a HTTPS connection to a webpage by entering https:// followed by URL in the browser address bar.
- Web browser initiates a connection to the web server. Use of https invokes the use of SSL protocol.
- An application, brows0065r in this case, uses the system port 443 instead of port 80 (used in case of http).
- The SSL protocol goes through a handshake protocol for establishing a secure session as discussed in earlier sections.
- The website initially sends its SSL Digital certificate to your browser. On verification of certificate, the SSL handshake progresses to exchange the shared secrets for the session.
- When a trusted SSL Digital Certificate is used by the server, users get to see a padlock icon in the browser address bar. When an Extended Validation Certificate is installed on a website, the address bar turns green.

- Once established, this session consists of many secure connections between the web server and the browser.

Use of HTTPS

- Use of HTTPS provides confidentiality, server authentication and message integrity to the user. It enables safe conduct of e-commerce on the Internet.
- Prevents data from eavesdropping and denies identity theft which is common attacks on HTTP.
- Present day web browsers and web servers are equipped with HTTPS support. The use of HTTPS over HTTP, however, requires more computing power at the client and the server end to carry out encryption and SSL handshake.

SECURE SHELL PROTOCOL (SSH):

The salient features of SSH are as follows −

- SSH is a network protocol that runs on top of the TCP/IP layer. It is designed to replace the TELNET which provided unsecure means of remote logon facility.
- SSH provides a secure client/server communication and can be used for tasks such as file transfer and e-mail.
- SSH2 is a prevalent protocol which provides improved network communication security over earlier version SSH1.

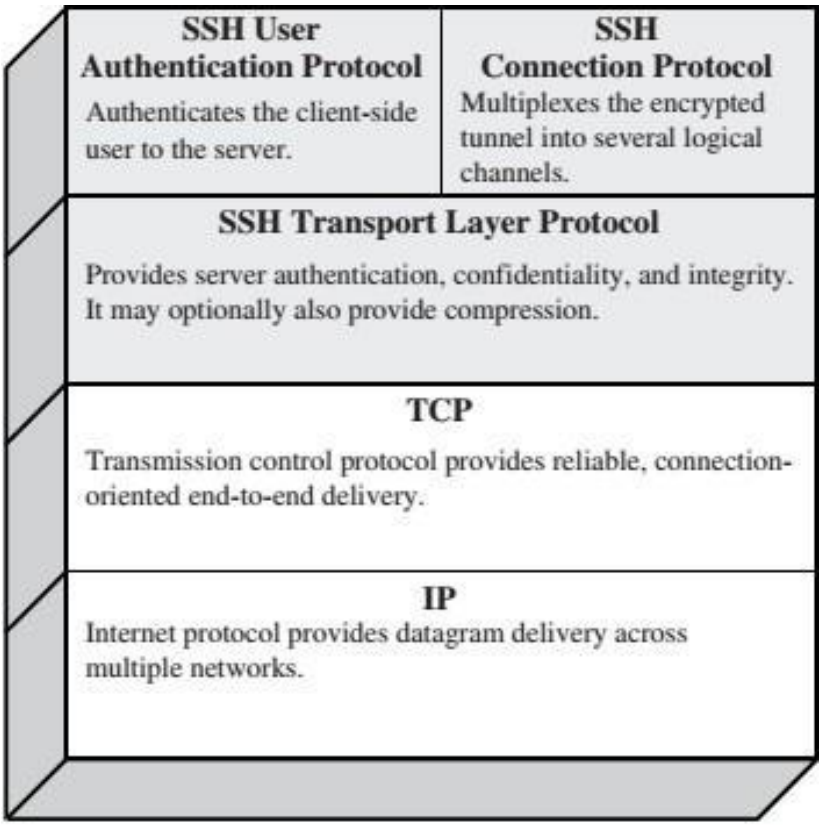


Figure 16.8 SSH Protocol Stack

SSH Services

SSH provides three main services that enable provision of many secure solutions. These services are briefly described as follows −

- **Secure Command-Shell (Remote Logon)** − It allows the user to edit files, view the contents of directories, and access applications on connected device. Systems administrators can remotely start/view/stop services and processes, create user accounts, and change file/directories permissions and so on. All tasks that are feasible at a machine's command prompt can now be performed securely from the remote machine using secure remote logon.

- **Secure File Transfer** − SSH File Transfer Protocol (SFTP) is designed as an extension for SSH-2 for secure file transfer. In essence, it is a separate protocol layered over the Secure Shell protocol to handle file transfers. SFTP encrypts both the username/password and the file data being transferred. It uses the same port as the Secure Shell server, i.e. system port no 22.
- **Port Forwarding (Tunneling)** − It allows data from unsecured TCP/IP based applications to be secured. After port forwarding has been set up, Secure Shell reroutes traffic from a program (usually a client) and sends it across the encrypted tunnel to the program on the other side (usually a server). Multiple applications can transmit data over a single multiplexed secure channel, eliminating the need to open many ports on a firewall or router.

Firewall

Firewall Design Principles

Firewall*: A firewall forms a barrier through which the traffic going in each direction must pass. A firewall security policy dictates which traffic is authorized to pass in each direction.*

A firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer.

Design goals for a firewall:

1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible, as explained later in this section.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass.

Various types of firewalls are used, which implement various types of security policies, as explained later in this section.

3. The firewall itself is immune to penetration.

Firewall Characteristics:

Major characteristics related to firewall protection are described below.

1. Various protection levels
2. Wireless network (Wi-fi) Protection
3. Internet and network access
4. Blockage against unauthorized access
5. Protection against malware
6. Provide access only to valid data packets
7. Provision of different configurations
8. Provision of numerous security policies

9. Allowing to pass authorized traffic that fulfils a set of rules
10. Firewall functions like an immune system for malware and unauthorized access; therefore, it ensures a secure system and an OS.

Access Control

Access Control policies are just one part of the Firewall Threat Defense (FTD) feature set that organizations use to control network traffic. As packets ingress the firewall, many checks occur. For example, is the packet part of an existing connection, and does the packet require decryption or network address translation? Once the packet has had these checks applied, it passes into the Access Control Policy (ACP).

An ACP can be assigned to one or more managed devices. However, a device can only have one ACP deployed at one time. The benefit of assigning a single ACP to more than one device is that a single change to the policy via the FMC UI can quickly be applied to multiple devices, reducing operational overheads.

Network data that is processed by managed devices can be filtered and controlled by a set of rules based on:

- Simple, easily determined transport and network layer characteristics: source and destination, ports,
  protocols and applications
- The latest contextual information on the traffic, including characteristics such as reputation, risk,
  business relevance, the application used, or URL visited
- Realm, user, user group, or ISE attribute
- Custom Security Group Tag (SGT)
- Characteristics of encrypted traffic; you can also decrypt this traffic for further analysis
- Whether unencrypted or decrypted traffic contains a prohibited file, detected malware, or intrusion
  attempt
- Time and day

Techniques that firewalls use to control access and enforce the site's security policy.

1. **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound.
2. **Direction control:** Determines the direction in which particular service requests may be

initiated and allowed to flow through the firewall.

3. **User control:** Controls access to a service according to which user is attempting to access it.

4. **Behavior control:** Controls how particular services are used.

Capabilities are within the scope of a firewall:

1. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks

2. A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.

3. A firewall can serve as the platform for IPSec.

Firewalls limitations

1. The firewall cannot protect against attacks that bypass the firewall.

2. The firewall does not protect against internal threats, such as an employee who unwittingly cooperates with an external attacker.

3. The firewall cannot protect against the transfer of virus-infected programs or files.

Types of Firewalls

There are three common types of firewalls:

1. Packet filters,
2. Application-level gateways,
3. Circuit-level gateways.

1. Packet-Filtering Router:

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet.

Filtering rules are based on information contained in a network packet:

Source IP address: The IP address of the system that originated the IP packet (e.g., 192.178.1.1)

Destination IP address: The IP address of the system the IP packet is trying to reach (e.g. 192.168.1.2)

Source and destination transport-level address: The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET .

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header.

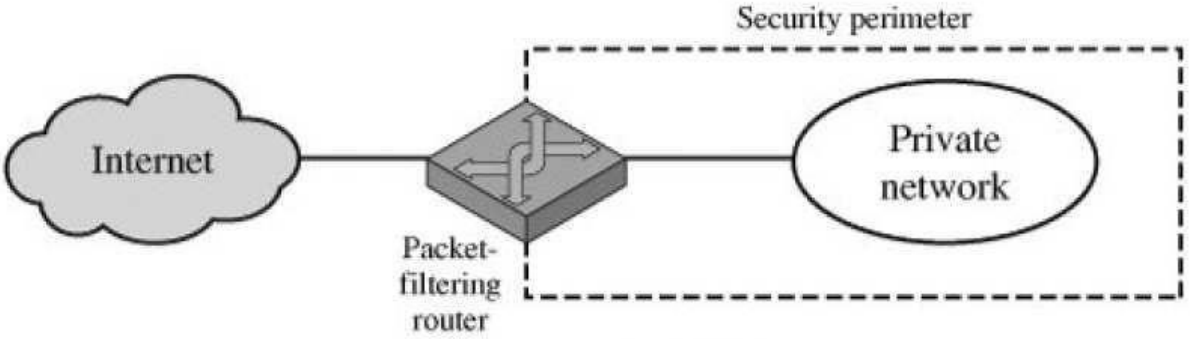
1. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet.

2. If there is no match to any rule, then a default action is taken.

3. Two default policies are possible:

   a. Default = *discard*: That which is not expressly permitted is prohibited.

   b. Default = *forward*: That which is not expressly prohibited is permitted.



(a) Packet-filtering router

Example:

A	**action**	**ourhost**	**port**	**theirhost**	**port**	**comment**	
	block	*	*	SPIGOT	*	we don't trust these people	
	allow	OUR-GW	25	*	*	connection to our SMTP port	
B	**action**	**ourhost**	**port**	**theirhost**	**port**	**comment**	
	block	*	*	*	*	default	
C	**action**	**ourhost**	**port**	**theirhost**	**port**	**comment**	
	allow	*	*	*	25	connection to their SMTP port	
D	**action**	**src**	**port**	**dest**	**port**	**flags**	**comment**
	allow	{our hosts}	*	*	25		our packets to their SMTP port
	allow	*	25	*	*	ACK	their replies
	action	**src**	**port**	**dest**	**port**	**flags**	**comment**
	allow	{our hosts}	*	*	*		our outgoing calls
E	allow	*	*	*	*	ACK	replies to our calls
	allow	*	*	*	>1024		traffic to nonservers

Weaknesses of packet filter firewalls:

IP address spoofing: The intruder transmits packets from the outside with a source IP address field containing an address of an internal host. The attacker hopes that the use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from

specific trusted internal hosts are accepted.

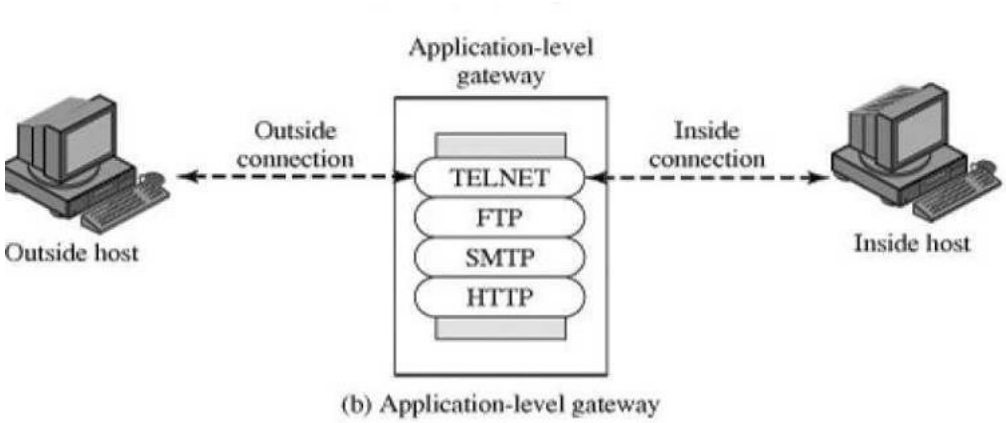
Stateful Firewall

A stateful inspection packet filter tightens up the rules for TCP traffic by creating a directory of outbound TCP connections

Source Address	Source Port	Destination Address	Destination Port	Connection State
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established

2. Application-Level Gateway:

- Application-level gateways tend to be more secure than packet filters
- An application-level gateway, also called a proxy server, acts as a relay of application-level traffic.
- The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed.
- When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host containing the application data between the two endpoints.
- If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall.



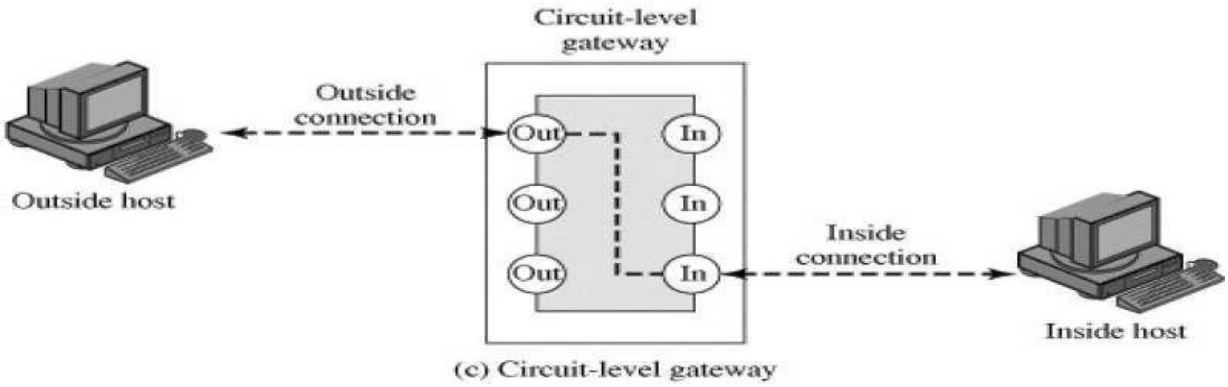
(b) Application-level gateway

3. Circuit-Level Gateway

- A typical use of circuit-level gateways is a situation in which the system administrator trusts the internal users.
- This can be a stand-alone system or it can be a specialized function performed by an application- level

gateway for certain applications.

- A circuit-level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outside host.
- Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents.
- The security function consists of determining which connections will be allowed.



(c) Circuit-level gateway

Bastion Host:

A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Typically, the bastion host serves as a platform for an application-level or circuit level gateway. Common characteristics of a bastion host include the following:

1. The bastion host hardware platform executes a secure version of its operating system, making ita trusted system.
2. Only the services that the network administrator considers essential are installed on the bastionhost. These include proxy applications such as Telnet, DNS, FTP, SMTP, and user authentication.
3. The bastion host may require additional authentication before a user is allowed access to theproxy services.

FIREWALL LOCATION AND CONFIGURATIONS

Firewalls are crucial components in network security, providing a barrier between a trusted internal network and untrusted external networks, such as the internet. Their location and configuration depend on the network architecture and security requirements. Here's a general overview:

Firewall Locations:

1. **Perimeter (Network Edge) Firewall:**
    - **Location:** Typically placed at the network perimeter between an organization's internal network and the internet.

- **Purpose:** Protects the entire internal network from external threats. It filters and monitors traffic entering and leaving the network.

2. **Internal Firewall:**
   - **Location:** Positioned within the internal network, dividing it into security zones.
   - **Purpose:** Provides additional segmentation and control within the internal network. It can restrict traffic between different parts of the internal network.
3. **Host-Based Firewall:**
   - **Location:** Installed on individual devices (e.g., computers, servers).
   - **Purpose:** Protects a specific device by monitoring and controlling incoming and outgoing network traffic based on an organization's configured security rules.

Firewall Configuration:

1. **Default Deny Rule:**
   - **Configuration:** Set a default rule to deny all traffic unless explicitly allowed. This ensures that only necessary and authorized traffic is permitted.
2. **Access Control Lists (ACLs):**
   - **Configuration:** Define ACLs based on IP addresses, ports, and protocols to control the flow of traffic. ACLs specify what traffic is allowed or denied.
3. **Stateful Inspection:**
   - **Configuration:** Enable stateful inspection to track the state of active connections. This allows the firewall to make context-aware decisions based on the current state of a connection.
4. **Proxy Services:**
   - **Configuration:** Use proxy services to inspect and filter application-layer traffic, providing an additional layer of security by analyzing the content of the traffic.
5. **Intrusion Detection and Prevention Systems (IDPS):**
   - **Configuration:** Integrate intrusion detection and prevention capabilities into the firewall to identify and block malicious activity.
6. **Virtual LANs (VLANs):**
   - **Configuration:** Utilize VLANs for network segmentation, and configure the firewall to control traffic between VLANs. This helps contain the impact of security incidents.
7. **Network Address Translation (NAT):**
   - **Configuration:** Implement NAT to hide internal IP addresses from external networks, enhancing security by obfuscating the internal network structure.
8. **Logging and Monitoring:**
   - **Configuration:** Enable logging for firewall events and regularly review logs for security incidents. Monitoring tools can provide real-time insights into network activity.
9. **Regular Updates:**
   - **Configuration:** Keep firewall firmware, software, and rule sets up to date to address security vulnerabilities and ensure optimal performance.
10. **Remote Access Policies:**
    - **Configuration:** If applicable, configure remote access policies to control and secure remote connections, such as Virtual Private Network (VPN) settings.
11. **User Authentication:**
    - **Configuration:** Implement user authentication to control access based on user credentials. This adds an extra layer of security beyond IP-based controls.

12. **Testing and Auditing:**
    - **Configuration:** Regularly test the firewall configuration and conduct security audits to identify and address potential vulnerabilities.
13. **Emergency Response Plan:**
    - **Configuration:** Have a plan in place to quickly reconfigure the firewall in response to security incidents. This may involve blocking specific IP addresses, adjusting rule sets, or implementing emergency measures.

Firewall configuration is highly dependent on the specific needs and risks of an organization. Regularly reviewing and updating the configuration based on evolving threats and network changes is essential for maintaining effective security.