



## UNIT-I

### COMPUTER SECURITY CONCEPTS:

The National Institute of Standards and Technology (NIST) define 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 unaffected manner, free from deliberate or inadvertent unauthorized manipulation of the system.

**Availability:** Availability of information refers to ensuring that authorized parties are able to access the information when needed.

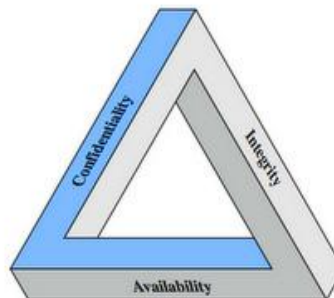


Figure: Figure: CIA Triad



---

## THE OSI SECURITY ARCHITECTURE

- The **Open Systems Interconnection (OSI) security architecture** provides a systematic framework for defining security attacks, mechanisms, and services.

### BASIC TERMINOLOGY:

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

**Attack:** A violation on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt to evade security services and violate the security policy of a system.

### ASPECTS OF SECURITY:

consider 3 aspects of information security:

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

### SECURITY ATTACKS:

Security attacks are classified into two:

- Passive attacks and
- Active attacks.

A passive attack attempts to learn or make use of information from the system but does not affect system resources.

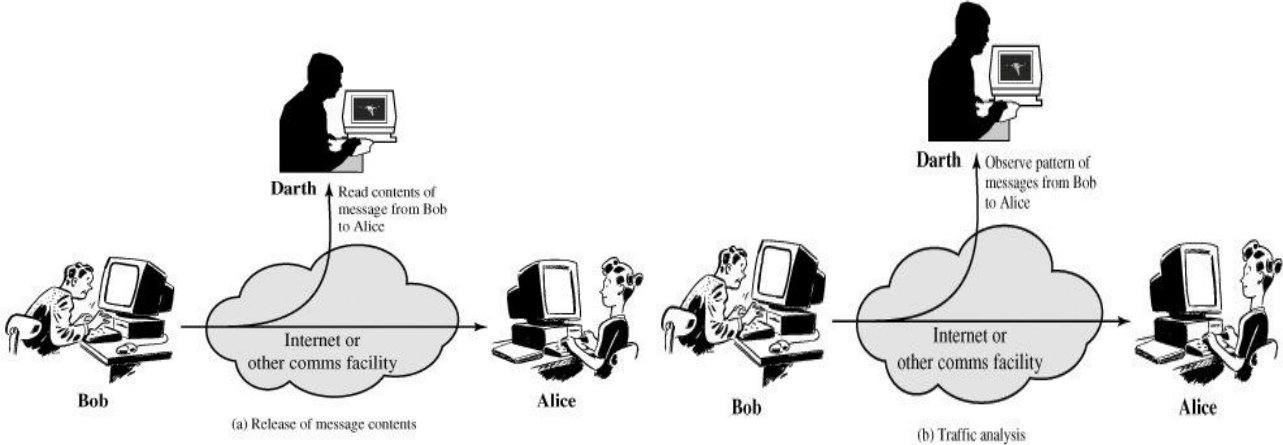
An active attack attempts to alter system resources or affect their operation.

#### Passive Attacks:

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted.

Two types of passive attacks are the **release of message contents** and **traffic analysis**.

- **Release of message contents:** The **release of message contents is easily understood**. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.



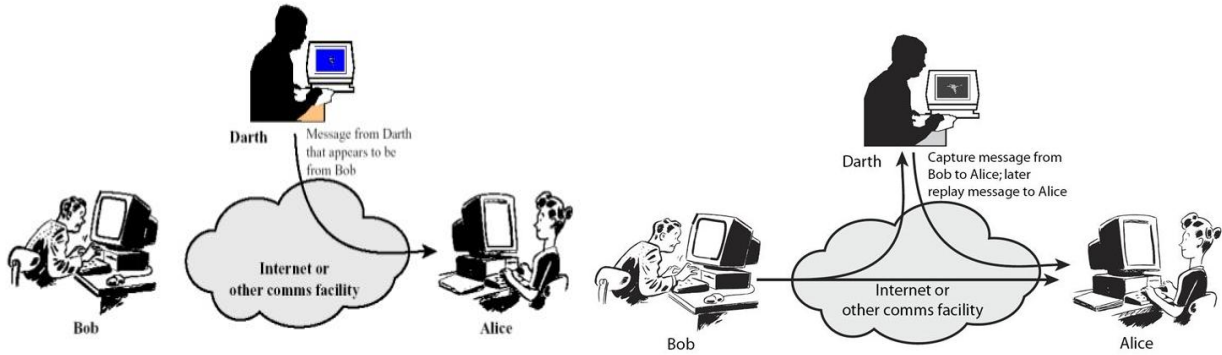
### TRAFFIC ANALYSIS:

- Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message.
- The common technique for masking contents is encryption.
- If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.
- This information might be useful in guessing the nature of the communication that was taking place.
- Passive attacks are very difficult to detect, because they do not involve any alteration of the data.
- Typically, the message traffic is sent and received in an apparently normal fashion, and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.
- However, it is feasible to prevent the success of these attacks, usually by means of encryption.
- Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

### Active Attacks:

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into **four** categories: **masquerade, replay, modification of messages, and denial of service**.

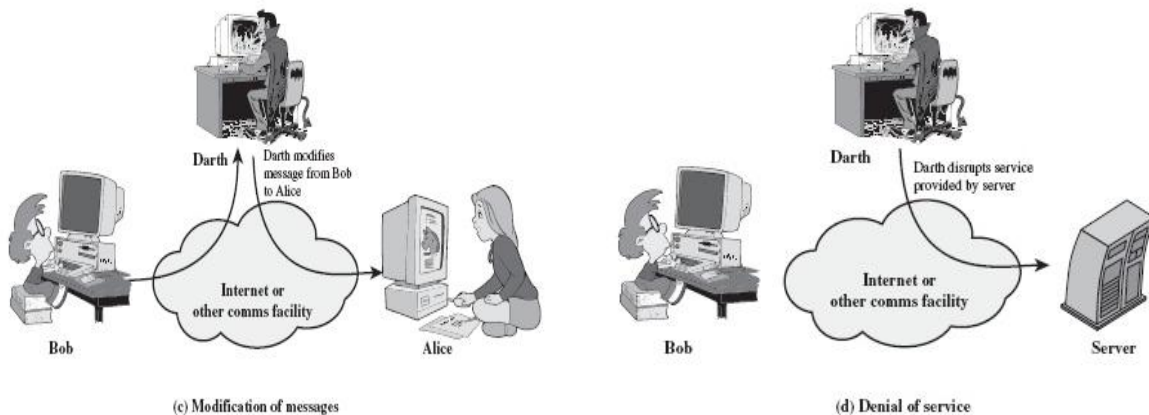
A **masquerade attack** is an attack that uses a fake identity, to gain unauthorized access to personal computer information through legitimate access identification. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.



**Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.**

**Modification of messages simply means that some portion of a valid message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.**

For example, a message meaning “Allow John Smith to read confidential file *accounts*” is *modified to mean* “Allow Fred Brown to read confidential file *accounts*.”



The **denial of service prevents the normal use or management of communications facilities**. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination. Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

### SECURITY SERVICES:

- Security service means a processing or communication service that is provided by a system to give a specific kind of protection to system resources.
- ❖ AUTHENTICATION
- ❖ ACCESS CONTROL



- 
- ❖ DATA CONFIDENTIALITY
  - ❖ DATA INTEGRITY
  - ❖ NONREPUDIATION
  - ❖ AVAILABILITY

**AUTHENTICATION:**

The authentication service is concerned with assuring that a communication is authentic. In the case of a **single message**, its function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an **ongoing interaction**, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Two specific authentication services are defined

- ❖ Peer entity authentication
- ❖ Data origin authentication

**Peer entity authentication:** Provides for the corroboration of the identity of a peer entities involved in communication. It is used for providing authentication at the time of connection establishment and during the process of data transmission.

**Data origin authentication:** Provides for the corroboration of the source of a data unit. It does not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail, where there are no prior interactions between the communicating entities.

**ACCESS CONTROL:**

The prevention of unauthorized use of resources. Access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

**DATA CONFIDENTIALITY:**

Confidentiality is the protection of transmitted data from passive attacks. The protection of data from unauthorized disclosure.

Types of confidentiality:

- ❖ **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).



---

### Types of integrity

- 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:**

It is assurance that someone cannot deny something. It is a method of guaranteeing message transmission between parties. 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.

### **AVAILABILITY:**

Availability is the method with assure the information and communications will be ready for use when excepted. Information is kept available to authorized persons when they need it. The availability can be significantly affected by a variety of attacks which are susceptible to authentication, encryption etc., whereas some attacks require physical action for preventing and recovering from the loss of availability

### **SECURITY MECHANISMS:**

Security mechanism is categorized into two types. They are,

- ❖ SPECIFIC SECURITY MECHANISMS
- ❖ PERVASIVE SECURITY MECHANISMS

### **SPECIFIC SECURITY MECHANISMS:**

**These mechanisms are** incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

- ❖ **Encipherment:** It refers to the process of applying 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 encryption keys.
- ❖ **Digital Signature:** Data appended to, or a cryptographic transformation of, a data unit must preserve the integrity of the data and prevents it from any unauthorized access.



- ❖ **Access Control:** A variety of mechanisms that enforce access rights to resources.
- ❖ **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- ❖ **Authentication Exchange:** A mechanism intended to ensure the identity of an entity by means of information exchange.
- ❖ **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.

### PERVASIVE SECURITY MECHANISMS:

Mechanisms that are not specific to any particular OSI security service or protocol layer.

- ❖ **Trusted Functionality:** That which is perceived to be correct with respect to some criteria.
- ❖ **Security Label:** the bounding value of a resource which specifies the security attributes associated with that resource.
- ❖ **Event Detection:** Detection of security-relevant events.
- ❖ **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.
- ❖ **Security Recovery:** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

### SOME BASIC TERMINOLOGY:

- ❖ An original message is known as the **plaintext**.
- ❖ The coded message is called the **ciphertext**.
- ❖ The process of converting from plaintext to ciphertext is known as **enciphering or encryption**.
- ❖ Restoring the plaintext from the ciphertext is **deciphering or decryption**.
- ❖ The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system or a cipher**.
- ❖ Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls “breaking the code.”
- ❖ The areas of cryptography and cryptanalysis together are called **cryptology**.

### SYMMETRIC CIPHER MODEL:

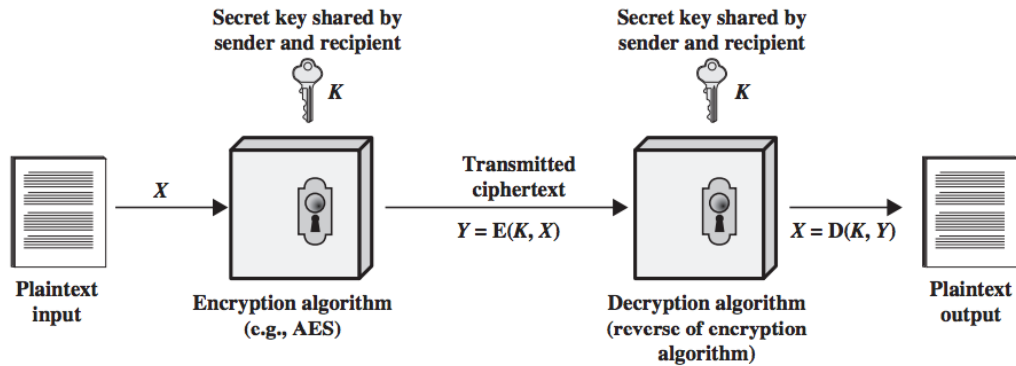
Symmetric encryption, also referred to as conventional encryption or single-key encryption

A symmetric encryption scheme has five ingredients.

- ❖ **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- ❖ **Encryption algorithm:** The encryption algorithm performs various substitutions and transformations on the plaintext.
- ❖ **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.



- ❖ **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- ❖ **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

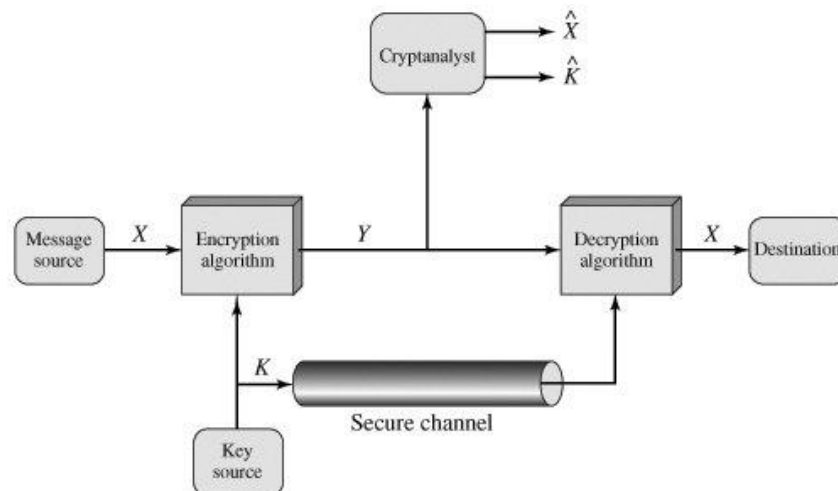


**Fig: simplified model of symmetric encryption**

### Requirements:

There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm
2. a secret key known only to sender / receiver: Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure



**Fig: model of symmetric encryption**





The essential elements of a symmetric encryption scheme, in the Figure. A source produces a message in plaintext,  $X = [X_1, X_2, \dots, X_M]$ . The elements of 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  $K = [K_1, K_2, \dots, K_J]$  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 and the encryption key as input, the encryption algorithm forms the ciphertext  $Y = [Y_1, Y_2, \dots, Y_N]$ . We can write this as  $Y = E(K, X)$ . This notation indicates that 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 (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate  $X^A$ . Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover by generating an estimate  $\check{K}$ .

### Cryptography:

Cryptographic systems are characterized along three independent dimensions:

1. **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as *product systems*, involve multiple stages of substitutions and transpositions.
2. **The number of keys used.** If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.
3. **The way in which the plaintext is processed.** A *block cipher* processes the input one block of elements at a time, producing an output block for each input block. A *stream cipher* processes the input elements continuously, producing output one element at a time, as it goes along.

### Cryptanalysis:

There are two general approaches to attacking a conventional encryption scheme:

- ❖ **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.
- ❖ **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.



Type of Attack	Known to Cryptanalyst
Ciphertext Only	<ul style="list-style-type: none"> <li>• Encryption algorithm</li> <li>• Ciphertext</li> </ul>
Known Plaintext	<ul style="list-style-type: none"> <li>• Encryption algorithm</li> <li>• Ciphertext</li> <li>• One or more plaintext–ciphertext pairs formed with the secret key</li> </ul>
Chosen Plaintext	<ul style="list-style-type: none"> <li>• Encryption algorithm</li> <li>• Ciphertext</li> <li>• Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key</li> </ul>
Chosen Ciphertext	<ul style="list-style-type: none"> <li>• Encryption algorithm</li> <li>• Ciphertext</li> <li>• Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key</li> </ul>
Chosen Text	<ul style="list-style-type: none"> <li>• Encryption algorithm</li> <li>• Ciphertext</li> <li>• Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key</li> <li>• Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key</li> </ul>

- ❖ An encryption scheme is **unconditionally Secure** if the ciphertext generated by the scheme does not contain enough information to determine uniquely the corresponding plaintext, no matter how much ciphertext is available. That is, no matter how much time an opponent has, it is impossible for him or her to decrypt the ciphertext simply because the required information is not there.
- ❖ An encryption scheme: **computationally secure** if The cost of breaking the cipher exceeds the value of information and the time required to break the cipher exceeds the lifetime of information

### SUBSTITUTION TECHNIQUES:

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

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

### Caesar Cipher:

- ❖ The earliest known, and the simplest, use of a substitution cipher was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example,

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



### Process of Caesar Cipher:

- ❖ In order to encrypt a plaintext letter, the sender positions the sliding ruler underneath the first set of plaintext letters and slides it to LEFT by the number of positions of the secret shift (here 3).
- ❖ The plaintext letter is then encrypted to the ciphertext letter on the sliding ruler underneath. The result of this process is depicted in the following illustration for an agreed shift of three positions.

Plaintext Alphabet	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
Ciphertext Alphabet	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
Ciphertext Alphabet	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
Plaintext Alphabet	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w

Let us assign a numerical equivalent to each letter:

a	b	c	d	e	f	g	h	i	j	k	l	m
0	1	2	3	4	5	6	7	8	9	10	11	12
n	o	p	q	r	s	t	u	v	w	x	y	z
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  $p$ , substitute the ciphertext letter  $C$

$$C = E(3, p) = (p + 3) \bmod 26$$

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

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

where  $k$  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. Following figure shows the results of applying this strategy to the example ciphertext. In this case, the plaintext leaps out as occupying the third line.

Three important characteristics of this problem enabled us to use a brute force cryptanalysis:

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try.
3. The language of the plaintext is known and easily recognizable.



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	ydrpc	rhc	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	hsjlg
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	vnnv	vn	jocna	cqn	cxpj	yjach
21	ummb	um	inbmz	bpm	bwoi	xizbg
22	tlla	tl	hmaly	aol	avnh	whyaf
23	skkz	sk	glzcx	znk	zumg	vgxze
24	rjyy	rj	fkyjw	ymj	ytlf	ufwyd
25	qiix	qi	ejxiv	xli	xske	tevxv

### MONOALPHABETIC CIPHERS:

- ❖ Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. 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. Before proceeding, we define the term *permutation*. A permutation of a finite set of elements  $S$  is an ordered sequence of all the elements of  $S$ , with each element appearing exactly once.

For example, if  $S = \{a, b, c\}$ , there are six permutations of  $S$ :

abc, acb, bac, bca, cab, cba

If the cryptanalyst knows the nature of the plaintext, then the analyst can exploit the regularities of the language.

```

UQZSOVUOHXMOPVGPZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPPQZWMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

```



As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English, such as is shown in Figure. If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match. A powerful tool is to look at the frequency of two-letter combinations, known as **digrams**.

The following table shows the frequency of letters in the above sentences

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

UZQSOVUOHXMOPVGPQZPEVSGZWSZOPFPESXUDBMETSXAIZ  
 t a e e t e a t h a t e e a a  
 VUEPHZHMDZSHZOWSFPAPPDTSVPPQUZWMXUZHUSX  
 e t t a t h a e e e a e t h t a  
 EPYEPOPDZSSZUFPPOMBZWPFPUPZHMDJUDTMOHMQ  
 e e e t a t e t h e t

Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.

#### Playfair Cipher:

The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair algorithm is based on the use of a 5 \* 5 matrix of letters constructed using a keyword.

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



In this case, the keyword is *monarchy*. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter. Plaintext is encrypted two letters at a time, according to the following rules:

1. **Repeating plaintext letters that are in the same pair are separated with a filler** letter, such as x, so that balloon would be treated as ba lx lo on.
2. **Two plaintext letters that fall in the same row of the matrix are each replaced** by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.
3. **Two plaintext letters that fall in the same column are each replaced by the letter beneath**, with the top element of the column circularly following the last. For example, mu is encrypted as CM.
4. **Otherwise, each plaintext letter in a pair is replaced by the letter that lies in** its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are  $26 * 26 = 676$  digrams, so that identification of individual digrams is more difficult. Despite this level of confidence in its security, the Playfair cipher is relatively easy to break, because it still leaves much of the structure of the plaintext language intact. A few hundred letters of ciphertext are generally sufficient.

### Hill Cipher:

Another interesting multiletter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. This encryption algorithm takes  $m$  successive plaintext letters and substitutes for them  $m$  ciphertext letters. The substitution is determined by  $m$  linear equations in which each character is assigned a numerical value ( $a = 0, b = 1, c, z = 25$ ).

For  $m = 3$ , the system can be described as

$$c_1 = (k_{11}p_1 + k_{21}p_2 + k_{31}p_3) \bmod 26$$

$$c_2 = (k_{12}p_1 + k_{22}p_2 + k_{32}p_3) \bmod 26$$

$$c_3 = (k_{13}p_1 + k_{23}p_2 + k_{33}p_3) \bmod 26$$

This can be expressed in terms of row vectors and matrices:<sup>6</sup>

$$(c_1 \ c_2 \ c_3) = (p_1 \ p_2 \ p_3) \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

$$\mathbf{C} = \mathbf{PK} \bmod 26$$

### Polyalphabetic Ciphers:

- ❖ Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**.



- ❖ The best known, and one of the simplest, such algorithm is referred to as the Vigenère cipher. In this scheme, the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers, with shifts of 0 through 25.

		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	a	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
	b	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	A
	c	C	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
	d	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
	e	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	D
	f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	z	Z	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

Table. The Modern Vigenère Tableau

key: *deceptive*

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

key:                    *deceptive**deceptive**deceptive*  
 plaintext:            wearediscoveredsaveyourself  
 ciphertext:            ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Expressed numerically, we have the following result.

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9



The periodic nature of the keyword can be eliminated by using a nonrepeating keyword that is as long as the message itself. Vigenère proposed what is referred to as an **autokey system**, in which a **keyword is concatenated with the plaintext itself** to provide a running key.

```
key:          deceptivewearediscoveredsav
plaintext:    wearediscoveredsaveyourself
ciphertext:   ZICVTWQNGKZEIIGASXSTSLVVWLA
```

### Vernam Cipher:

The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918.

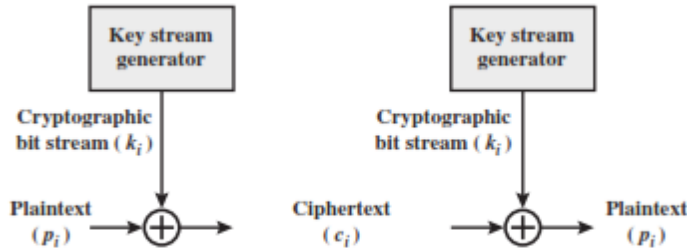


Figure 2.7 Vernam Cipher

### One-Time Pad:

Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad, is unbreakable**.

An example should illustrate our point. Suppose that we are using a Vigenère scheme with 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message. Consider the cipher text.

ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

We now show two different decryptions using two different keys:

```
ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        pxlmvmsydofoyrvzwc tnlbnecvgdupahfzzlmnyih
plaintext:  mr mustard with the candlestick in the hall
```

```
ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
key:        mfugpmiydgaxgoufhklllmhsqdgogtewbqfgyovuhwt
plaintext:  miss scarlet with the knife in the library
```





The one-time pad offers complete security but, in practice, has two fundamental difficulties:

1. **There is the practical problem of making large quantities of random keys.** Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.
2. **Even more daunting is the problem of key distribution and protection.** For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

### Transposition Techniques:

A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

#### **RAIL FENCE TECHNIQUE:**

The simplest such cipher is the **rail fence technique, in which the plaintext is** written down as a sequence of diagonals and then read off as a sequence of rows.

For example, to encipher the message “meet me after the toga party” with a rail fence of depth 2 we write the following:

```

m e m a t r h t g p r y
e t e f e t e o a a t

```

The encrypted message is

**MEMATRHTGPRYETEFETEOAAT**

#### **Row Transposition Ciphers:**

A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm.

```

Key:           4 3 1 2 5 6 7
Plaintext:    a t t a c k p
               o s t p o n e
               d u n t i l t
               w o a m x y z
Ciphertext:   TTNAAPTMTSUOAODWCOIXKNLYPETZ

```



### **PHISHING DEFENSIVE MEASURES:**

- ❖ Phishing is a fraudulent process, which attempts to acquire sensitive information, such as usernames, passwords, and credit card numbers by masquerading as a trustworthy entity in an electronic communication.
- ❖ **Spear-phishing emails** have a high success rate because they mimic messages from an authoritative source, such as a financial institution, a communications company, or some other easily recognizable entity with a reputable brand.
- ❖ In general, these phishing techniques are manifested in social engineering, URL/Link manipulation, filter evasion e.g., using images to hide malicious links, and website forgery.
- ❖ **Web Forgery**, also known as Phishing, is a form of identity theft that occurs when a malicious website impersonates a valid one in order to obtain someone's sensitive information.
- ❖ Pharming is another technique intended to redirect a website's traffic to another, fake site.
- ❖ Pharming can be conducted either by changing the hosts file on a victim's computer or by exploitation of a vulnerability in DNS server software.

### **SAFE BROWSING TOOL:**

Since the web is the most frequently used attack vector, it is important to have protection for browsers, especially when a search is used.

- ❖ **The Web of Trust (WOT) Plugin for Safe Browsing:** The WOT is a community-based collection of websites, based on a reputation achieved through the ratings of millions of users. It is a free safe surfing plugin for major browsers and provides website ratings and reviews to help web users as they search, surf and shop online.
- ❖ WOT uses color-coded symbols to show the reputation of a site: Green indicates the site is trusted by the community, yellow warns a user to be cautious and red indicates potential

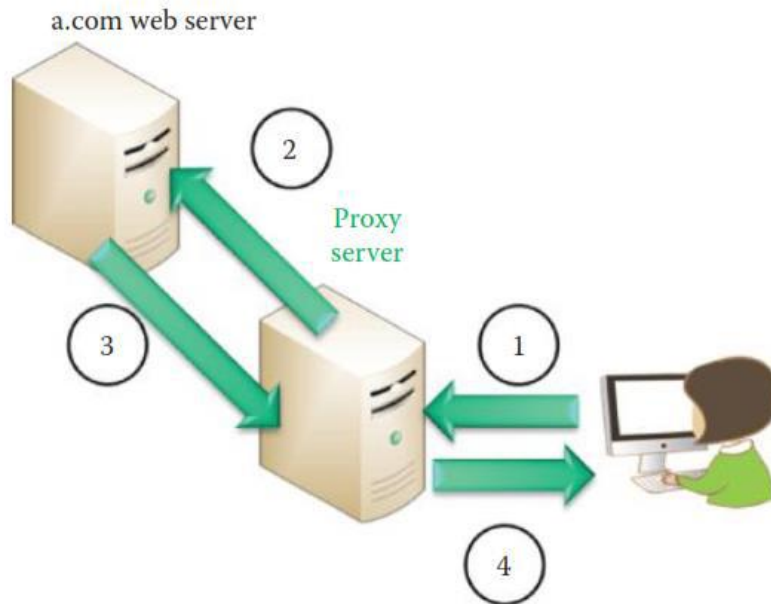
### **WEB-BASED ATTACKS:**

The vulnerabilities in web-based attacks are manifested in a variety of ways. For example, the inadequate validation of user input may occur in one of the following attacks: **Cross-Site Scripting (XSS or CSS), HTTP Response Splitting or SQL Injection.**

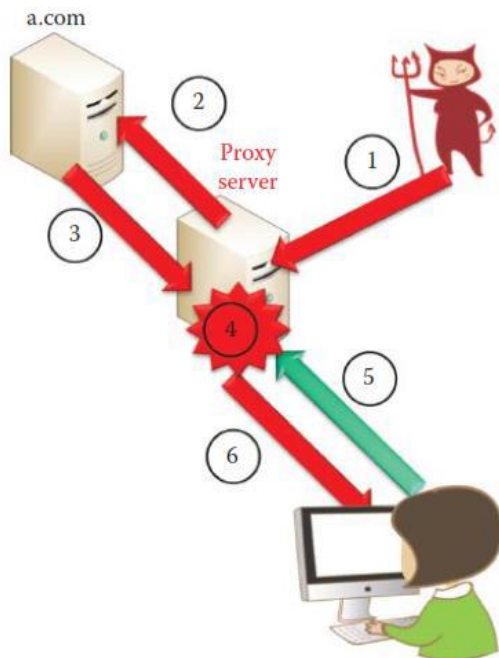
### **HTTP RESPONSE SPLITTING ATTACKS:**

- ❖ HTTP response splitting occurs when:
  - Data enters a web application through an untrusted source, most frequently an HTTP request.
  - The data is included in an HTTP response header sent to a web user without being validated for malicious characters.
- ❖ At its root, the attack is straightforward: an attacker passes malicious data to a vulnerable application, and the application includes the data in an HTTP response header.
- ❖ HTTP response splitting attacks may happen where the server script embeds user data in HTTP response headers without appropriate sanitation.
- ❖ This typically happens when the script embeds user data in the redirection URL of a redirection response (HTTP status code 3xx), or when the script embeds user data in a cookie value or name when the response sets a cookie.

- ❖ Attacker uses a web server, which has a vulnerability enabling HTTP response splitting, and a proxy/cache server in a HTTP response splitting attack.
- ❖ HTTP response splitting is the attacker's ability to send a single HTTP request that forces the web server to form an output stream, which is then interpreted by the target as two HTTP responses instead of one response.



**FIGURE:** A normal operation for executing a Redirect Script for language preference and the response is cached in a proxy server.





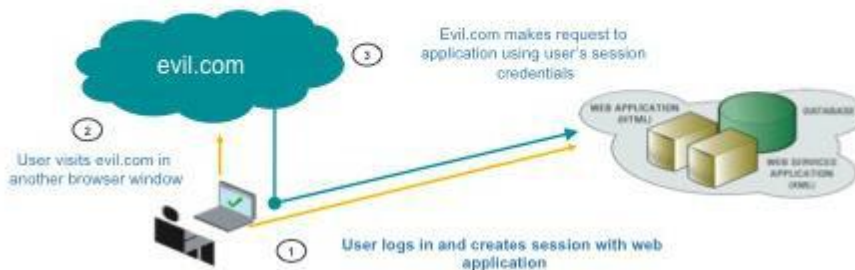
**FIGURE:** Attacker uses a.com web server, which has a vulnerability enabling HTTP response splitting, and a proxy/cache server in a HTTP response splitting attack. A victim will retrieve the cached second response when accessing the a.com.

### **CROSS-SITE REQUEST FORGERY (CSRF OR XSRF):**

Cross-Site Request Forgery (CSRF) is an attack that forces an end user to execute unwanted actions on a web application in which they're currently authenticated. CSRF attacks specifically target state-changing requests, not theft of data, since the attacker has no way to see the response to the forged request. With a little help of social engineering (such as sending a link via email or chat), an attacker may trick the users of a web application into executing actions of the attacker's choosing. If the victim is a normal user, a successful CSRF attack can force the user to perform state changing requests like transferring funds, changing their email address, and so forth. If the victim is an administrative account, CSRF can compromise the entire web application.

#### **Cross Site Request Forgery Attacks**

Attacking trust relationships:



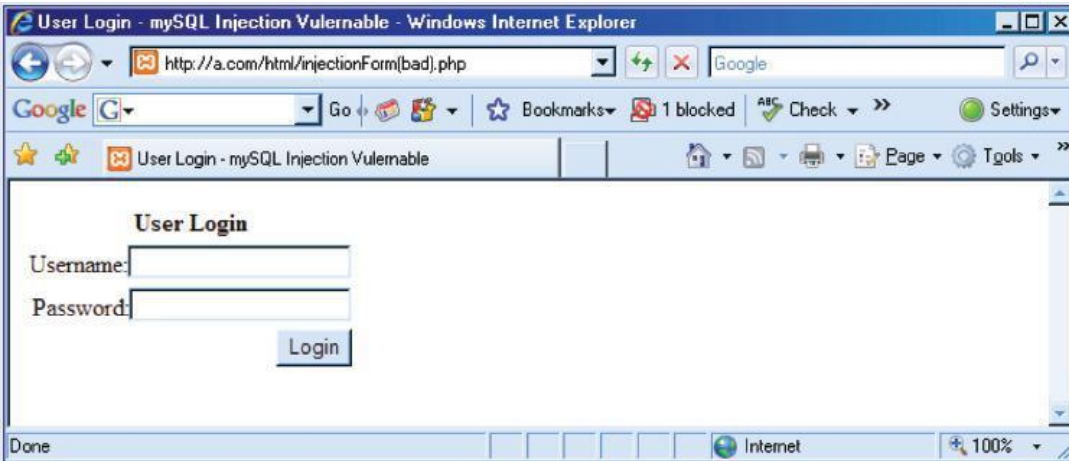
### **DATABASE DEFENSIVE MEASURES:**

#### **STRUCTURED QUERY LANGUAGE (SQL) INJECTION ATTACKS:**

A SQL injection attack consists of insertion or "injection" of a SQL query via the input data from the client to the application. A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system. SQL injection attacks are a type of injection attack, in which SQL commands are injected into data-plane input in order to affect the execution of predefined SQL commands.

#### **The Manner in Which to Execute a SQL Injection Attack:**

As an example of a SQL injection attack, consider the normal user login request shown in Figure (a). A user supplies their username and password, and this SQL query checks to see if the user/password combination is in the database. The query is of the form  
 \$query = "SELECT username,password FROM login WHERE username = '\$username' AND password = '\$password'";



### **Buffer Overflow:**

A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data Storage area) than it was intended to hold. Since buffers are created to contain a finite amount of data, the extra information - which has to go somewhere - can overflow into adjacent buffers, corrupting or overwriting the valid data held in them. It may occur accidentally through programming error; buffer overflow is an increasingly common type of security attack on data integrity.

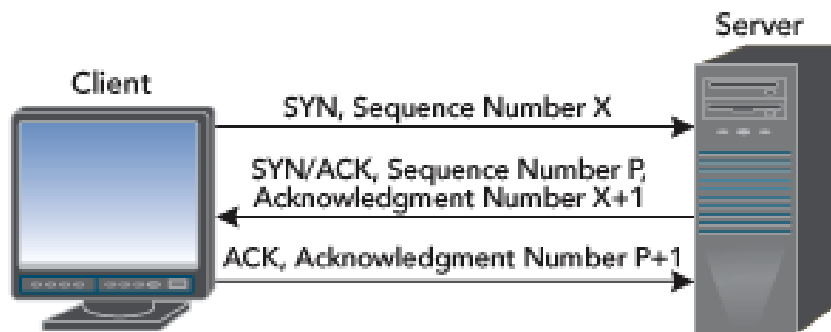
The stack is a section of memory used for temporary storage of information. In a stack-based buffer overflow attack, the attacker adds more data than expected to the stack, overwriting data. For example, "Let's say that a program is executing and reaches the stage where it expects to use a postal coder or zip code, which it gets from a Web-based form that customers filled Out. " The longest postal code is fewer than twelve characters, but on the web form, the attacker typed in the letter "A" 256 times, followed by Some other commands. The data overflows the buffer allotted for the zip code and the attacker's commands fall into the stack. After a function is called, the address of the instruction following the function call is pushed onto the stack to be saved so that the function knows where to return control when it is finished.



Figure : stack representation of a normal stack

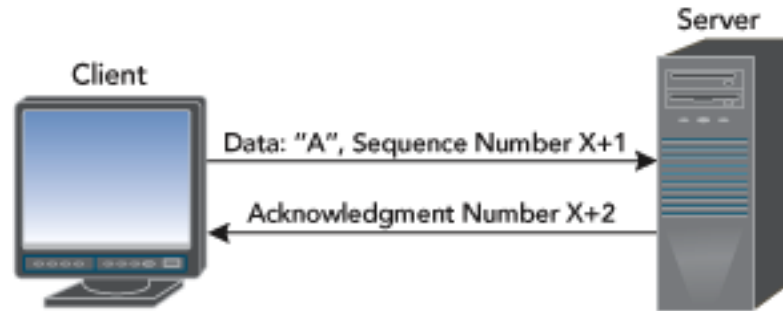
### **TCP session hijacking:**

TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. In order to guarantee that packets are delivered in the right order, TCP uses acknowledgement (ACK) packets and sequence numbers to create a “full duplex reliable stream connection between two end points,” with the end points referring to the communicating hosts. The connection between the client and the server begins with a three-way handshake.



**Figure(a): TCP Three-Way Handshake**

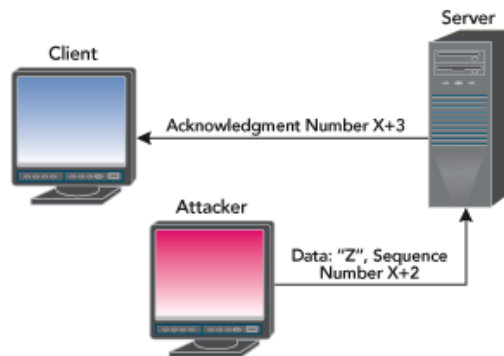
For now, observe what happens to these sequence numbers when the client starts sending data to the server (see **Figure (b)**). In order to keep the example simple, the client sends the character A in a single packet to the server.



**Figure(b) Sending Data over TCP**

TCP Session hijacking is when a hacker takes over a TCP session between two machines. Since most authentications only occur at the start of a TCP session, this allows the hacker to gain access to a machine.

A popular method is using source-routed IP packets. This allows a hacker at point A on the network to participate in a conversation between B and C by encouraging the IP packets to pass through its machine. If source-routing is turned Off, the hacker can use "blind" hijacking see figure (c), whereby it guesses the responses of the two machines. Thus, the hacker can send a command, but can never see the However, a common command would be to set a password allowing access from somewhere else on the net. A hacker can also be "inline" between B and C using a sniffing program to watch the conversation. This is known as a "**man-in-the-middle attack**".

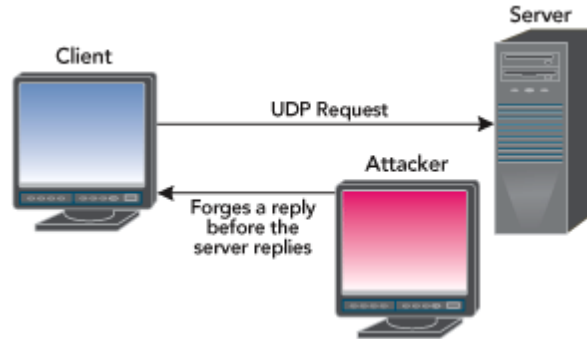


**Figure(c) Blind hijacking**

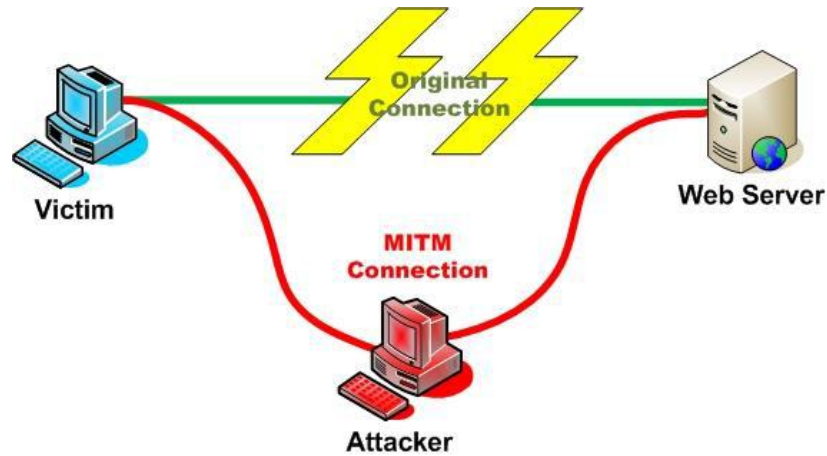
A common component of such an attack is to execute a denial -of-service attack against one end-point to stop it from responding. This attack can be either against the machine to force it to crash, or against the network connection to force heavy packet loss. TCP session hijacking is a much more complex and difficult attack.

### **UDP Hijacking:**

UDP which stands for User Datagram Protocol is defined as a connectionless protocol. It offers a direct way to send and receive datagram's over an IP network. UDP doesn't use sequence numbers like TCP. It is mainly used for broadcasting messages across the network or for doing DNS queries. Hijacking a session over a User Datagram Protocol (UDP) is exactly the same as over TCP, except that UDP attackers do not have to worry about the overhead of managing sequence numbers and other TCP mechanisms. Since UDP is connectionless, injecting data into a session without being detected is extremely easy.

**Man in the Middle Attacks:**

In cryptography, a man-in-the-middle attack (MITM) is an attack in which an attacker is able to read, insert and modify at will, messages between two parties without either party knowing that the link between them has been compromised. The attacker must be able to observe and intercept messages going between the two victims.





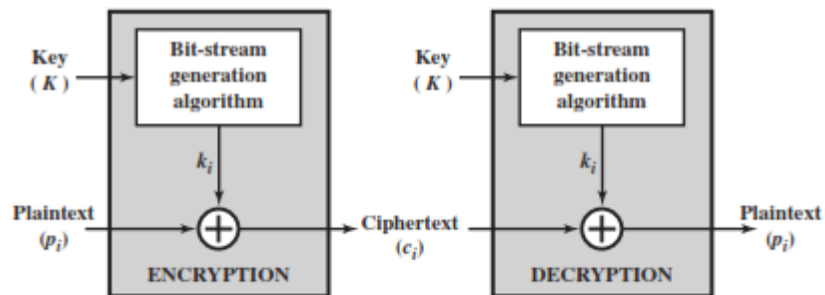


## UNIT-II

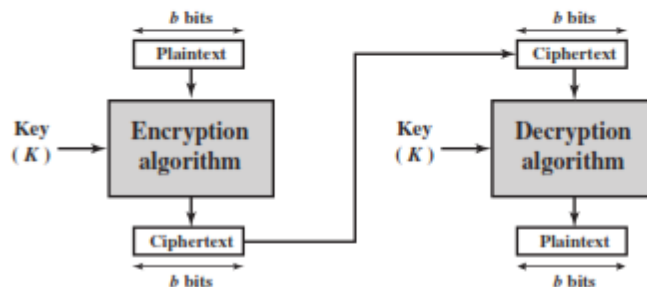
### Stream Ciphers and Block Ciphers:

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the auto keyed Vigenère cipher and the Vernam cipher.

- ❖ A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- ❖ Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a symmetric encryption key



(a) Stream cipher using algorithmic bit-stream generator



(b) Block cipher

### The Feistel Cipher:

Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of  $k$  bits and a block length of  $n$  bits, allowing a total of  $2^k$  possible transformations, rather than the  $2^n!$  transformations available with the ideal block cipher.

In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:



- **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

**FEISTEL CIPHER STRUCTURE:** The left-hand side of Figure depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length  $2w$  bits and a key. The plaintext block is divided into two halves,  $L_0$  and  $R_0$ . The two halves of the data pass through  $n$  rounds of processing and then combine to produce the ciphertext block. Each round  $i$  has as inputs  $L_{i-1}$  and  $R_{i-1}$  derived from the previous round, as well as a subkey  $K_i$  derived from the overall  $K$ . In general, the subkeys  $K_i$  are different from  $K$  and from each other.

All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a round function  $F$  to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey  $K_i$ .

**Permutation** is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.
- **Key size:** Larger key size means greater security but may decrease encryption/decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

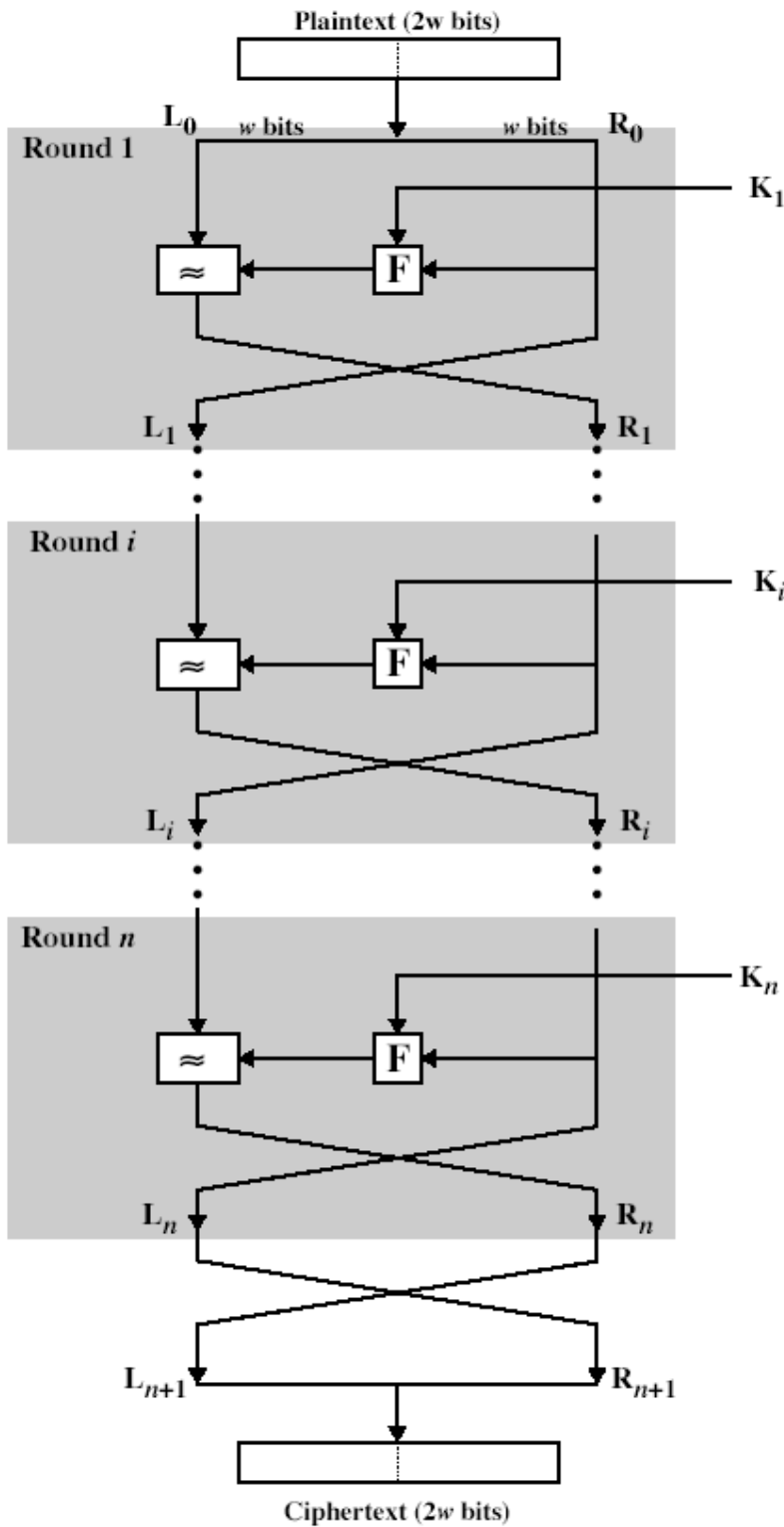


Fig: Feistel Cipher structures



- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function F:** Again, greater complexity generally means greater resistance to cryptanalysis.

There are two other considerations in the design of a Feistel cipher:

- **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality.

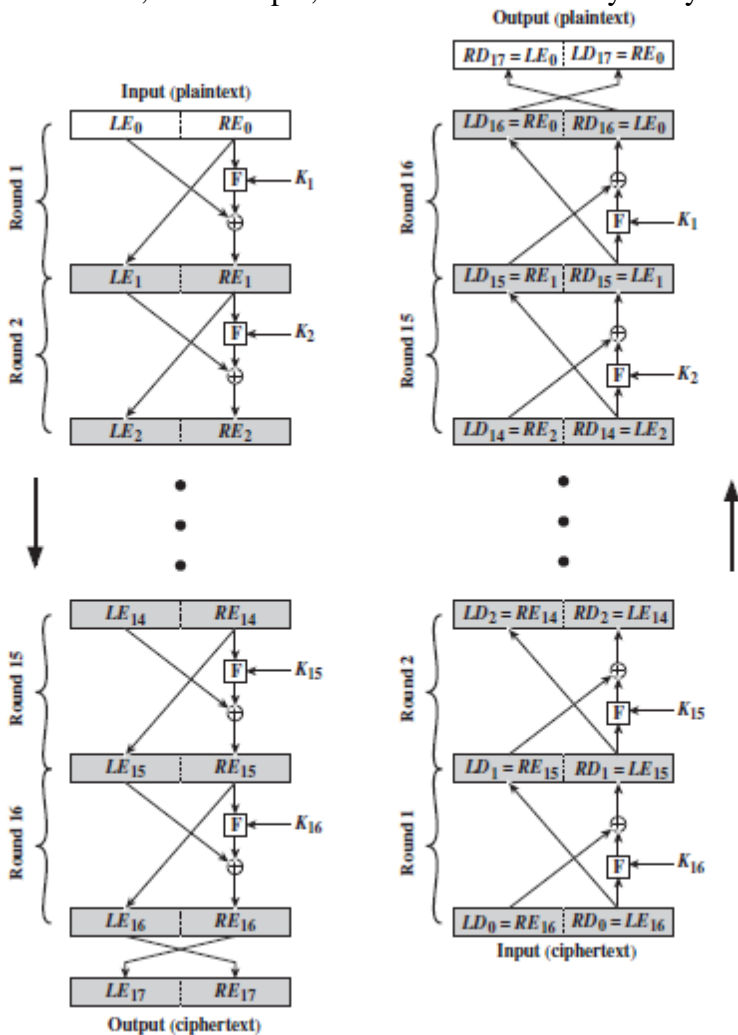


Figure 3.3 Feistel Encryption and Decryption (16 rounds)



### Feistel Decryption Algorithm:

The process of decryption with a Feistel cipher is essentially the same as the encryption process.

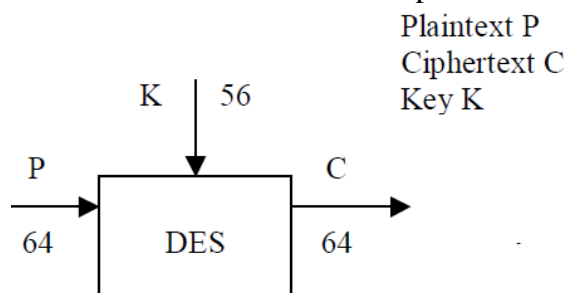
The rule is as follows:

- Use the ciphertext as input to the algorithm, but use the subkeys  $K$  in reverse order.
- That is, use  $K_n$  in the first round,  $K_{n-1}$  in the second round, and so on until  $K$  is used in the last round. This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption.
- To see that the same algorithm with a reversed key order produces the correct result, which shows the encryption process going down the left-hand side and the decryption process going up the right-hand side for a 16-round algorithm.
- For clarity, we use the notation  $LE_i$  and  $RE_i$  for data traveling through the encryption algorithm and  $LD_i$  and  $RD_i$  for data traveling through the decryption algorithm.
- The diagram indicates that, at every round, the intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped.

encryption process $LE_{16} = RE_{15}$ $RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$ <b>in general</b> $LE_i = RE_{i-1}$ $RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$	On the decryption side, $LD_1 = RD_0 = LE_{16} = RE_{15}$ $RD_1 = LD_0 \oplus F(RD_0, K_{16})$ $= RE_{16} \oplus F(RE_{15}, K_{16})$ $= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16})$
---	--

### Data Encryption Standard:

- DES is a Symmetric-key algorithm for the encryption of electronic data.
- DES originated at IBM in 1977 & was adopted by the U.S Department of Defence. Now it is under the NIST (National Institute of Standard & Technology)
- Data Encryption Standard (DES) is a widely-used method of data encryption using a private (secret) key.
- DES applies a 56-bit key to each 64-bit block of data. The process can run in several modes and involves 16 rounds or operations.





### Overall structure

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher.

Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*.
- This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**.
- Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher,

The right-hand portion of below shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a *subkey ( $K_i$ ) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.*

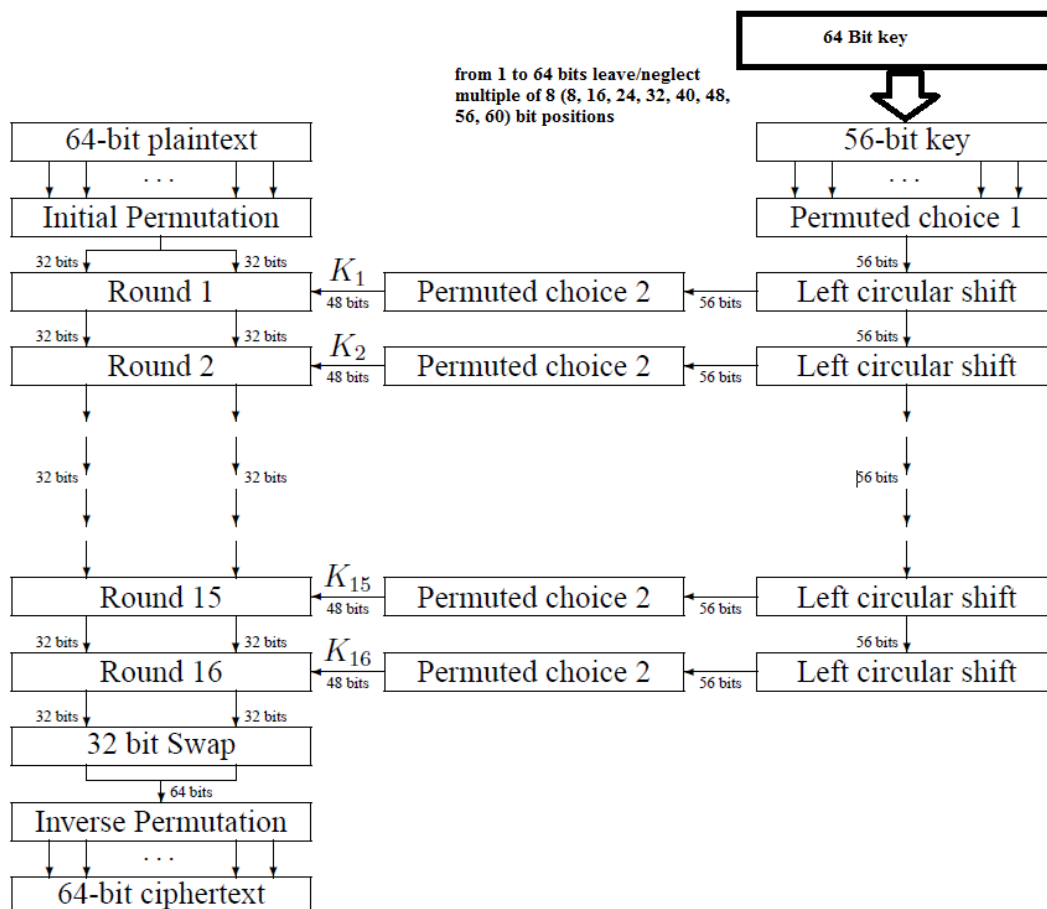


Figure : Flow Diagram of DES algorithm for encrypting data.



Table 3.2 Permutation Tables for DES

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP<sup>-1</sup>)

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	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	30	31	32	1

(d) 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

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

$M_1$   $M_2$   $M_3$   $M_4$   $M_5$   $M_6$   $M_7$   $M_8$   
 $M_9$   $M_{10}$   $M_{11}$   $M_{12}$   $M_{13}$   $M_{14}$   $M_{15}$   $M_{16}$   
 $M_{17}$   $M_{18}$   $M_{19}$   $M_{20}$   $M_{21}$   $M_{22}$   $M_{23}$   $M_{24}$   
 $M_{25}$   $M_{26}$   $M_{27}$   $M_{28}$   $M_{29}$   $M_{30}$   $M_{31}$   $M_{32}$   
 $M_{33}$   $M_{34}$   $M_{35}$   $M_{36}$   $M_{37}$   $M_{38}$   $M_{39}$   $M_{40}$   
 $M_{41}$   $M_{42}$   $M_{43}$   $M_{44}$   $M_{45}$   $M_{46}$   $M_{47}$   $M_{48}$   
 $M_{49}$   $M_{50}$   $M_{51}$   $M_{52}$   $M_{53}$   $M_{54}$   $M_{55}$   $M_{56}$   
 $M_{57}$   $M_{58}$   $M_{59}$   $M_{60}$   $M_{61}$   $M_{62}$   $M_{63}$   $M_{64}$



Where  $M_i$  is a binary digit. Then the permutation  $X = (IP(M))$  is as follows:

$M_{58}$	$M_{50}$	$M_{42}$	$M_{34}$	$M_{26}$	$M_{18}$	$M_{10}$	$M_2$
$M_{60}$	$M_{52}$	$M_{44}$	$M_{36}$	$M_{28}$	$M_{20}$	$M_{12}$	$M_4$
$M_{62}$	$M_{54}$	$M_{46}$	$M_{38}$	$M_{30}$	$M_{22}$	$M_{14}$	$M_6$
$M_{64}$	$M_{56}$	$M_{48}$	$M_{40}$	$M_{32}$	$M_{24}$	$M_{16}$	$M_8$
$M_{57}$	$M_{49}$	$M_{41}$	$M_{33}$	$M_{25}$	$M_{17}$	$M_9$	$M_1$
$M_{59}$	$M_{51}$	$M_{43}$	$M_{35}$	$M_{27}$	$M_{19}$	$M_{11}$	$M_3$
$M_{61}$	$M_{53}$	$M_{45}$	$M_{37}$	$M_{29}$	$M_{21}$	$M_{13}$	$M_5$
$M_{63}$	$M_{55}$	$M_{47}$	$M_{39}$	$M_{31}$	$M_{23}$	$M_{15}$	$M_7$

If we then take the inverse permutation

$$Y = IP^{-1}(X) = IP^{-1}(IP(M)), \text{ it can be}$$

seen that the original ordering of the bits is restored.

### DETAILS OF SINGLE ROUND

Below figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

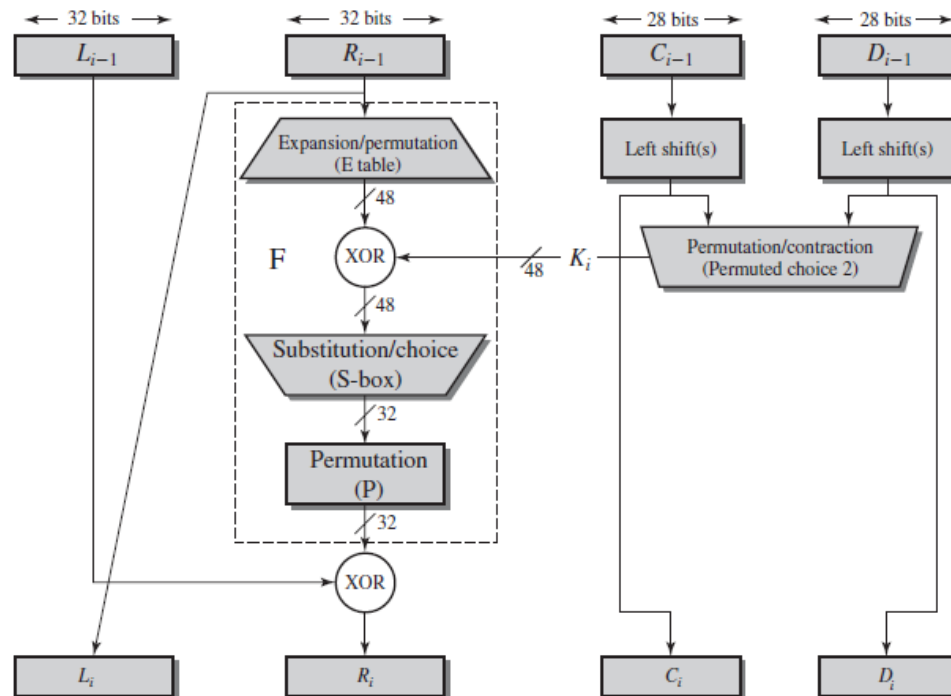


Figure :Single Round of DES Algorithm





The round key  $K_i$  is 48 bits. The  $R$  input is 32 bits. This  $R$  input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the  $R$  bits (Table 3.2c). The resulting 48 bits are XORed with  $K_i$ . This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table (d). The role of the S-boxes in the function  $F$  is illustrated in Figure 3.7. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box  $S_i$  form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output.

For example, in  $S_1$ , for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001. Each row of an S-box defines a general reversible substitution. Figure 3.2 may be useful in understanding the mapping. The figure shows the substitution for row 0 of box  $S_1$ . The operation of the S-boxes is worth further comment. Ignore for the moment the contribution of the key ( $K_i$ ). If you examine the expansion table, you see that the 32 bits of input are split into groups of 4 bits and then become groups of 6 bits by taking the outer bits from the two adjacent groups. For example, if part of the input word is

... efgh ijkl mnop ...

This becomes ... defghi hijklm lmnopq ...

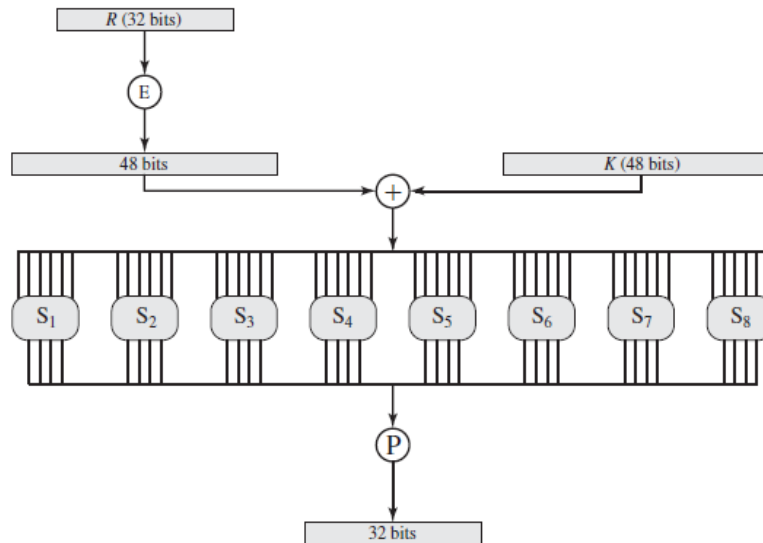


Figure 3.7 Calculation of  $F(R, K)$

### **DES DECRYPTION:**

Whatever process we following in the encryption that process is used for decryption also but the order of key is changed on input message (cipher text).

Reverse order of keys are  $K_{16}, K_{15}, \dots, K_1$ .

**The Avalanche Effect:**

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.
- This is referred to as the avalanche effect.

**THE STRENGTH OF DES:****The Use of 56-Bit Keys:**

With a key length of 56 bits, there are 256 possible keys, which is approximately  $7.2 \times 10^{16}$ . A brute-force attack appears impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher. Diffie and Hellman postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond. This would bring the average search time down to about 10 hours.

**The Nature of the DES Algorithm:**

- Possibilities of cryptanalysis is done by finding the characteristics of DES algorithm.
- Learning of S-Box logic is complex.
- Weakness of the S-boxes not been discovered.

**Timing Attacks:**

- A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.
- A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
- DES appears to be fairly resistant to a successful timing attack.

**Block Cipher Design Principles:**

There are three critical aspects of block cipher design: **the number of rounds, design of the function F, and key scheduling.**

**Number of Rounds:**

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack. This criterion was certainly used in the design of DES.

**Design of Function F:**

- The heart of a Feistel block cipher is the function F, which provides the element of confusion in a Feistel cipher. Thus, it must be difficult to “unscramble” the substitution performed by F.
- F must be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be.

**Key Schedule Algorithm:**

- With any Feistel block cipher, the key is used to generate one subkey for each round.
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.

**Triple DES (3DES):**

- Triple DES is simply another mode of DES operation. It takes three 64-bit keys, for an overall key length of 192 bits.
- The Triple DES then breaks the user provided key into three subkeys, padding the keys if necessary so they are each 64 bits long.
- The procedure for encryption is exactly the same as regular DES, but it is repeated three times. Hence the name Triple DES. The data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key.

**ADVANCED ENCRYPTION STANDARD(AES):**

- The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001.
- AES is a block cipher intended to replace DES for commercial applications.
- It uses a 128-bit block size and a key size of 128, 192, or 256 bits.

**AES parameters:**

Key size(words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round Key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

**Inner Workings of a Round**

The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of its counterpart in the encryption algorithm. The four stages are as follows:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following:

1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns



Again, the tenth round simply leaves out the **Inverse Mix Columns** stage. Each of these stages will now be considered in more detail.

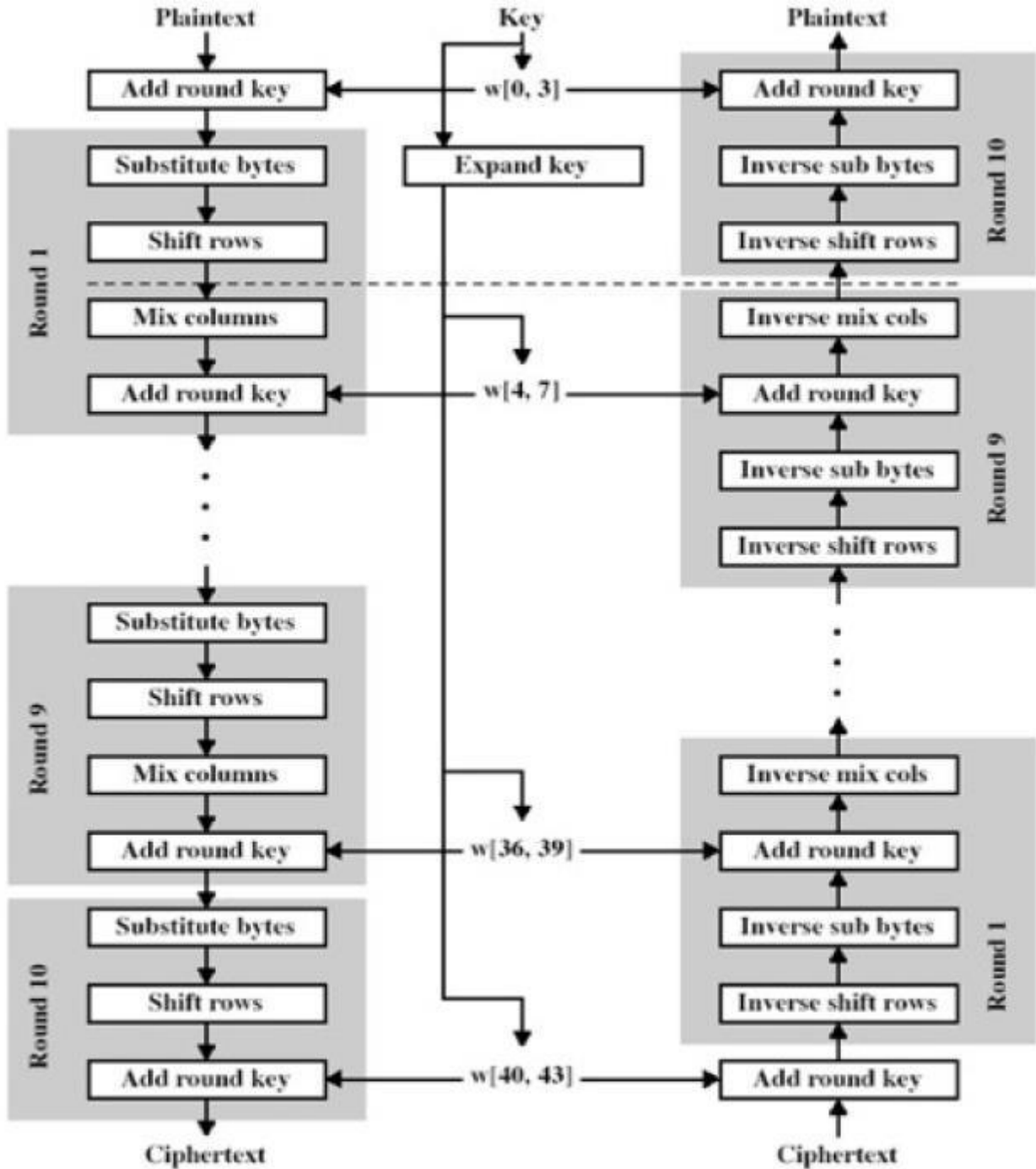
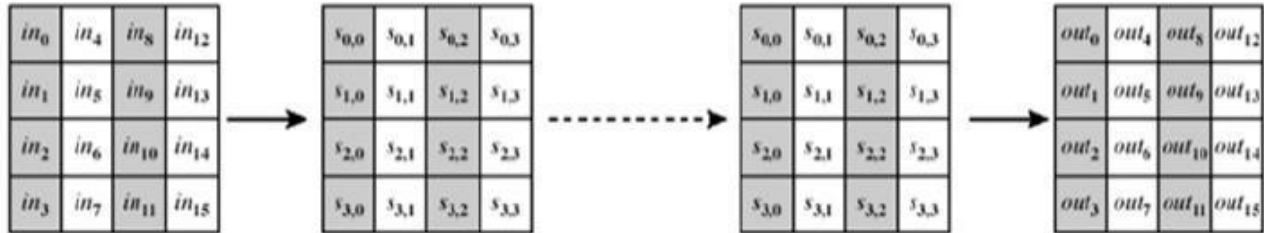


FIGURE: overall structure of the AES algorithm

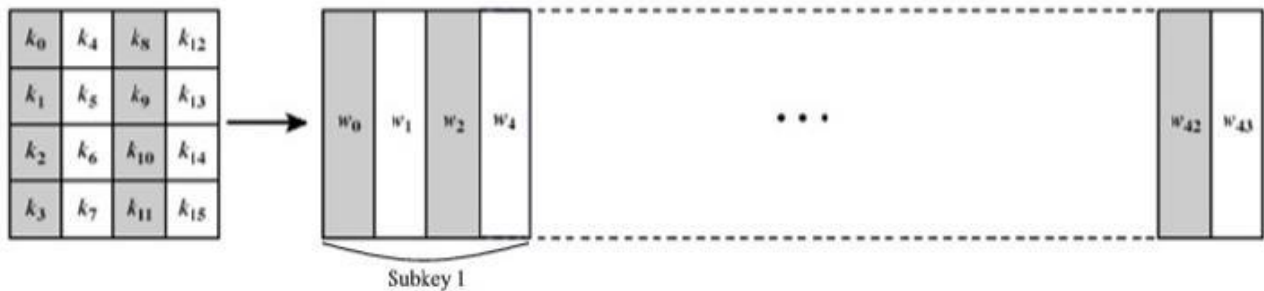


### Substitute Bytes

This stage (known as SubBytes) is simply a table lookup using a  $16 \times 16$  matrix of byte values called an s-box. This matrix consists of all the possible combinations of an 8-bit sequence ( $2^8 = 16 \times 16 = 256$ ). However, the s-box is not just a random permutation of these values and there is a well-defined method for creating the s-box tables. The designers of Rijndael showed how this was done unlike the s-boxes in DES for which no rationale was given.



(a) Input, state array, and output



(b) Key and expanded key

### Figure 7.2: Data structures in the AES algorithm.

Again the matrix that gets operated upon throughout the encryption is known as **state**. We will be concerned with how this matrix is effected in each round. For this particular round each byte is mapped into a new byte in the following way: the leftmost nibble of the byte is used to specify a particular row of the s-box and the rightmost nibble specifies a column. For example, the byte {95} (curly brackets represent hex values in FIPS PUB 197) selects row 9 column 5 which turns out to contain the value {2A}.

This is then used to update the **state** matrix. Figure 7.3 depicts this idea.

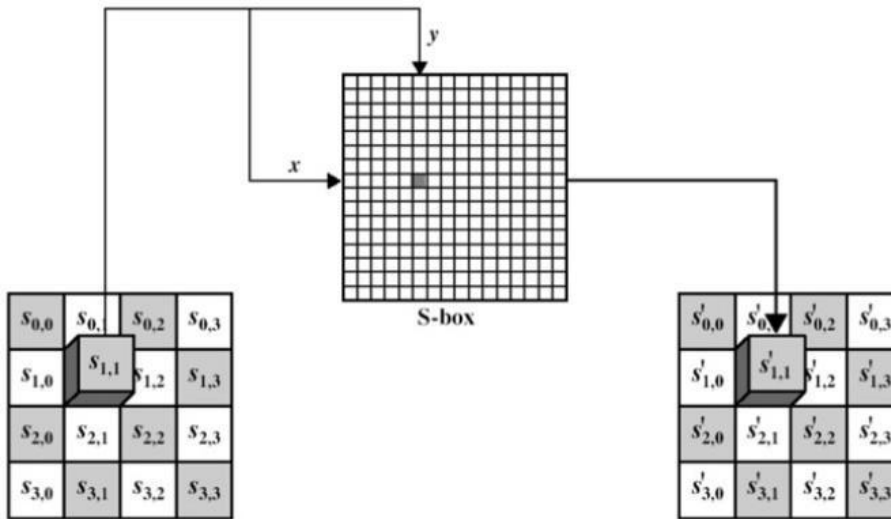


Figure 7.3: Substitute Bytes Stage of the AES algorithm.

**Shift Rows Transformation:**

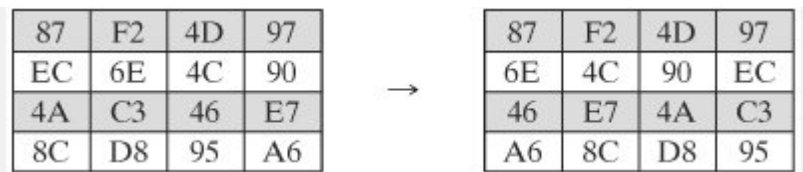
Shift row transformation are two types.

- Forward Shift row transformation which is used in encryption.
- Inverse Shift row transformation which is used in decryption.

**FORWARD SHIFT ROW TRANSFORMATION:**

- The first row of State matrix is not altered.
- For the second row, a 1-byte circular left shift is performed.
- For the third row, a 2-byte circular left shift is performed.
- For the fourth row, a 3-byte circular left shift is performed.

The following is an example of ShiftRows:

**INVERSE SHIFT ROWS:**

- Performs the circular shifts in the opposite direction for each of the last three rows, with a one-byte circular right shift for the second row and soon.

**MIX COLUMNS TRANSFORMATION:**

Mix columns transformation are two types.

- Forward Mix columns transformation which is used in encryption.
- Inverse Mix columns transformation which is used in decryption.

**Forward Mix columns transformation:**

- Forward Mix columns transformation called mix columns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all 4 bytes in that column. The transformation can be defined by the following matrix multiplication on state.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

**inverse Mix columns transformation:**

- The inverse mix column transformation, called InvMixColumns, is defined by the following matrix multiplication:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

**AddRoundKey Transformation:**

- In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key.
- The inverse add round key transformation is identical to the forward add round key transformation, because the XOR operation is its own inverse.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

 $\oplus$ 

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

 $=$ 

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

**AES Key Expansion:**

- The 128-bit key value can be expanded into 44 words i.e. 44X32=1408bits
- In each round 4 words will be used i.e. 4x32=128 bits
- In Addroundkey first 4 words w0,w1,w2,w3 are used.
- In first round,w4,w5,w6,w7 are used and soon.

The 128 bit key is expanded as follows

- First 128 bit key is arranged as a 4x4 matrix each value size is 8-bits
- The first 32 bits (k0,k1,k2,k3) is considered as w0.
- The first 32 bits (k4,k5,k6,k7) is considered as w1.
- The first 32 bits (k8,k9,k10,k11) is considered as w2.
- The first 32 bits (k12,k13,k14,k15) is considered as w4.
- Next 4 words w4,w5,w6,w7 are followed as

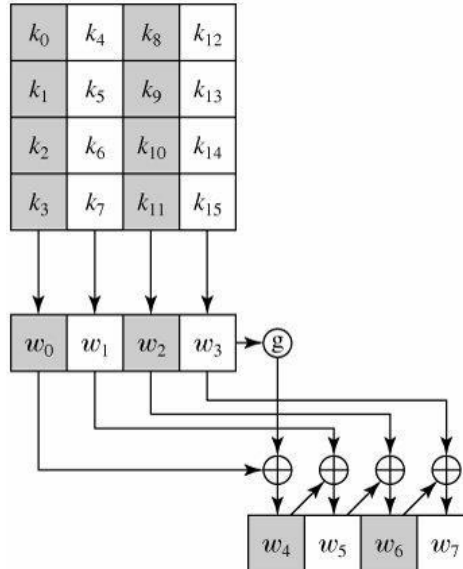


$$w_4 = w_0 \oplus w_3$$

$$w_5 = w_1 \oplus w_4$$

$$w_6 = w_2 \oplus w_5$$

$$w_7 = w_3 \oplus w_6$$



### **BLOWFISH:**

- Blow fish is a symmetric block cipher developed by Bruce Schneier in year 1993.
- Blow fish is designed to have following characteristics
- Speed: Blowfish encrypts data on 32-bit microprocessor at a rate of 18 clock cycles per byte.
- Compact: it can run in less than 5k memory.
- Simple: very easy to implement.
- Variably secure: the key length is variable and can be as long as 448 bits. This allows a trade-off between higher speed and higher security.
- Blowfish is a Feistel type model.

### **BLOWFISH ALGORITHM:**

- Blowfish is Feistel type model, iterating a simple encryption function 16 times.
- Blowfish block size is 64 & key can be up to 448 bits.
- Blow fish encryption 64-bit blocks of plaintext into 64-bit block of cipher.
- Blow fish make use of a key that ranges from 32 bits to 448 bits (one to fourteen 32-bit keys).
- The keys are stored in a k-array (one to 14 32 bits)

$K_1, K_2, \dots, K_j$  where  $1 \leq j \leq 14$ .





### Encryption and Decryption

Blowfish uses two primitive operations:

- Addition: Addition of words, denoted by +, is performed modulo 232.
- Bitwise exclusive-OR: This operation is denoted by  $\oplus$

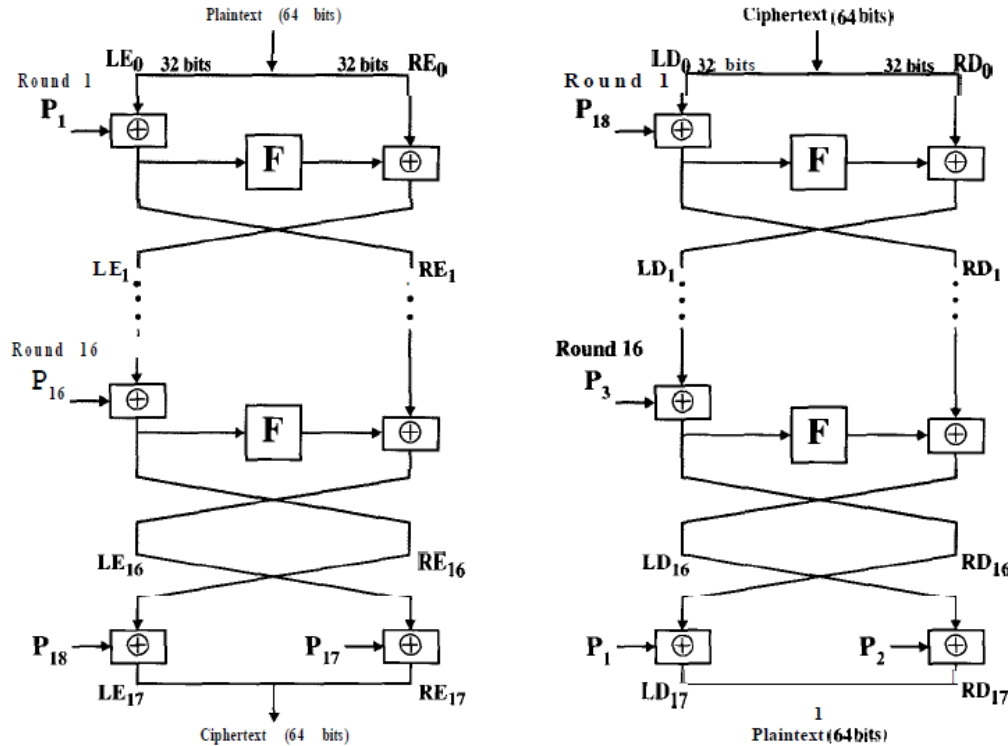


Figure Blowfish Encryption and Decryption.

The function F is shown in below Figure. The 32-bit input to F is divided into 4 bytes. If we label those bytes a, b, c, and d, then the function can be defined as follows:

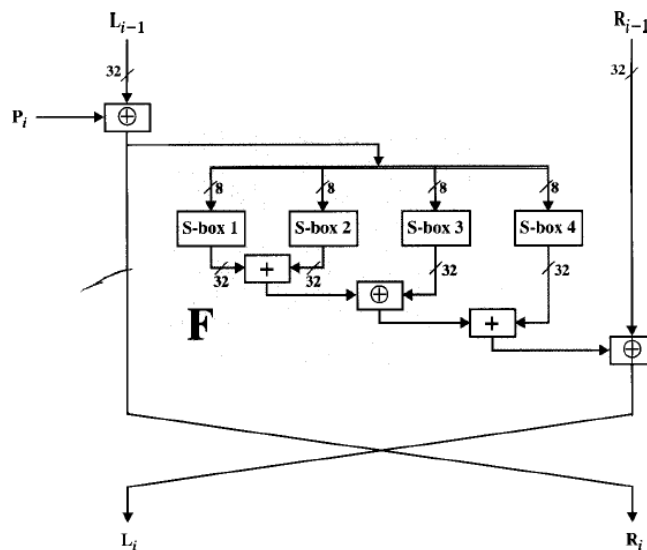


Figure Detail of Single Blowfish Round.



$$F[a, b, c, d,] = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d}$$

**Blowfish Decryption:**

Blowfish decryption occurs in the same algorithmic direction as encryption. Rather than the reverse. The algorithm can be defined as follows:

for  $i = 1$  to 16 do

$$RD_i = LD_{i-1} \oplus P_{19-i};$$

$$LD_i = F[RD_i] \oplus RD_{i-1};$$

$$LD_{i+1} = RD_i \oplus P_i;$$

$$RD_{i+1} = LD_i \oplus P_{i+1};$$

**CAST-128:**

- It is an encryption algorithm.
- It takes 64-bit plain text, 128 bit key as input and produces 64-bit cipher text as output. It has 16 rounds.

**Description of Algorithm:**

CAST-128 belongs to the class of encryption algorithms known as Feistel ciphers; overall operation is thus similar to the Data Encryption Standard (DES). The full encryption algorithm is given in the following four steps.

INPUT: plaintext  $m_1 \dots m_{64}$ ;

key  $K = k_1 \dots k_{128}$ .

OUTPUT: ciphertext  $c_1 \dots c_{64}$ .

1. (key schedule) Compute 16 pairs of subkeys  $\{K_{mi}, K_{ri}\}$  from  $K$
2. Split the plaintext into left and right 32-bit halves  $L_0 = m_1 \dots m_{32}$  and  $R_0 = m_{33} \dots m_{64}$ .
3. It has 16 rounds for  $i$  from 1 to 16, compute  $L_i$  and  $R_i$  as follows:

$$L_i = R_{i-1};$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_{mi}, K_{ri}), \text{ where } f \text{ is function (} f \text{ is of Type 1, Type 2, or Type 3, depending on } i \text{).}$$

4.  $c_1 \dots c_{64} \leftarrow (R_{16}, L_{16})$ . (Exchange final blocks  $L_{16}, R_{16}$  and concatenate to form the ciphertext.)

Decryption is identical to the encryption algorithm given above, except that the rounds (and therefore the subkey pairs) are used in reverse order to compute  $(L_0, R_0)$  from  $(R_{16}, L_{16})$ .

**Pairs of Round Keys:**

- CAST-128 uses a pair of subkeys per round: a 32-bit quantity "Km" is used as a "masking" key and a 5-bit quantity "Kr" is used as a "rotation" key.

**Non-Identical Rounds:**

Three different round functions are used in CAST-128.

The rounds are as follows

- where "D" is the data input to the  $f$  function and "Ia" - "Id" are the most significant byte through least significant byte of  $I$ , respectively).



- All functions use the operation "+" and "-" are addition and subtraction  $\oplus$  XOR, and "<<<" is the circular left-shift operation.

Type 1:  $I = ((K_{mi} + D) \lll K_{ri})$

$f = ((S1[Ia] \oplus S2[Ib]) - S3[Ic]) + S4[Id]$

Type 2:  $I = ((K_{mi} \oplus D) \lll K_{ri})$

$f = ((S1[Ia] - S2[Ib]) + S3[Ic]) \oplus S4[Id]$

Type 3:  $I = ((K_{mi} - D) \lll K_{ri})$

$f = ((S1[Ia] + S2[Ib]) \oplus S3[Ic]) - S4[Id]$

Rounds 1, 4, 7, 10, 13, and 16 use f function Type 1.

Rounds 2, 5, 8, 11, and 14 use f function Type 2.

Rounds 3, 6, 9, 12, and 15 use f function Type 3.

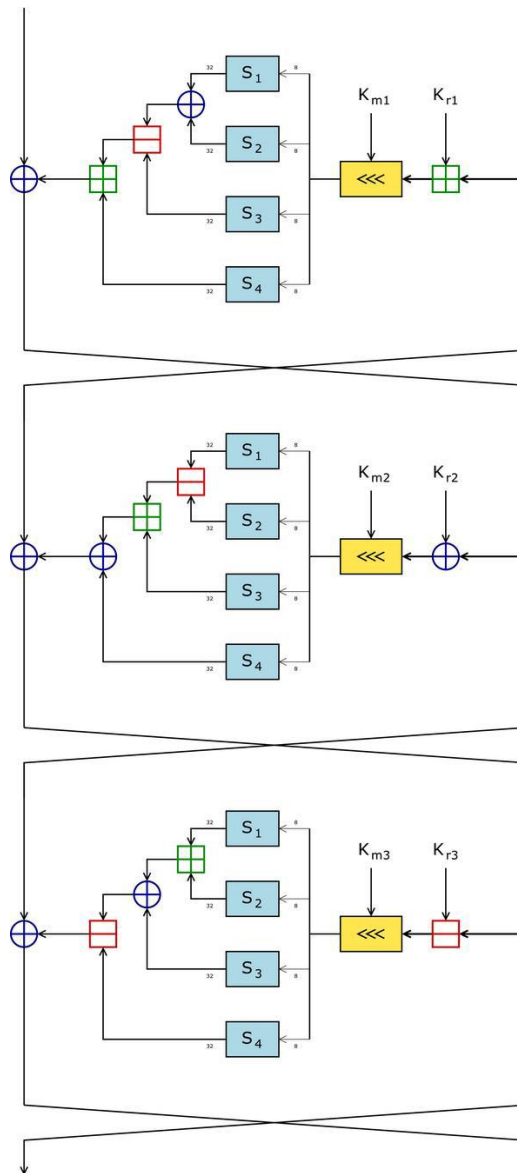


Figure: Three rounds of the CAST-128 block cipher

## INTERNATIONAL DATA ENCRYPTION ALGORITHM(IDEA):

- IDEA (International Data Encryption Algorithm) was originally called IPES (Improved Proposed Encryption Standard).
- It was developed by Xuejia Lai and James L. Massey of ETH Zurich.
- IDEA was designed to be efficient to compute in software. It encrypts a 64-bit block of plaintext into a 64-bit block of ciphertext using a 128-bit key.
- It was published in 1991, so cryptanalysts have had time to find weaknesses.
- IDEA is similar to DES in some ways.
- Both of them operate in rounds, and both have a complicated mangler function that does not have to be reversible in order for decryption to work.
- Instead, the mangler function is run in the same direction for encryption as decryption, in both IDEA and DES.
- In fact, both DES and IDEA have the property that encryption and decryption are identical except for key expansion.
- With DES, the same keys are used in the reverse order
- with IDEA, the encryption and decryption keys are related in a more complex manner.

### Primitive Operations

- Each primitive operation in IDEA maps two 16-bit quantities into a 16-bit quantity.
- IDEA uses three operations
- $\oplus$ -XOR,
- $+$ -Addition all easy to compute in software, to create a mapping.

$\oplus$  Multiplication Operation.

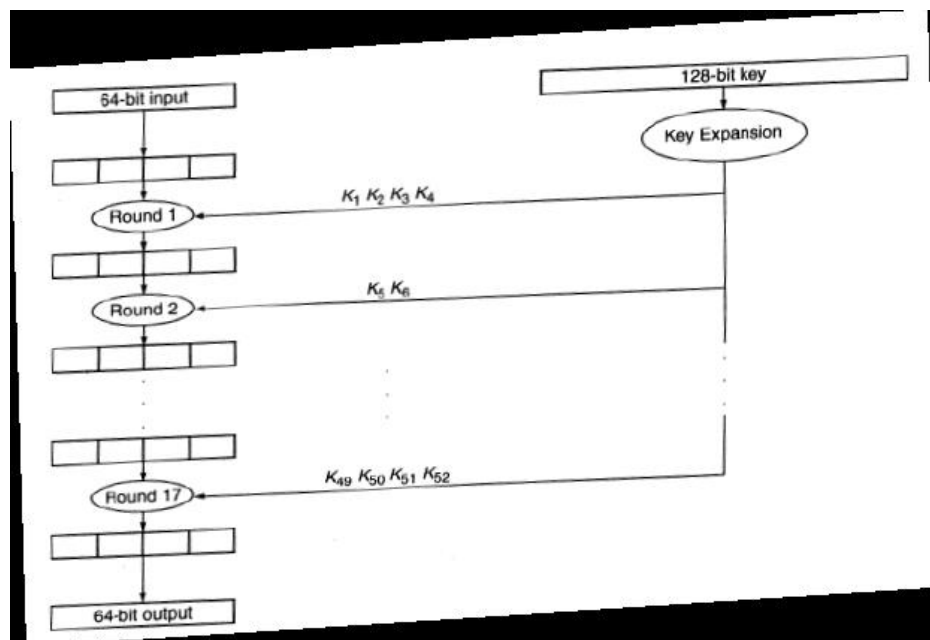


Figure: basic structure of IDEA

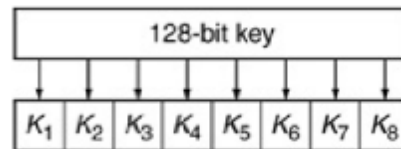


It has total 17 rounds

- In IDEA, Odd rounds accepts 4 subkeys.
- Even rounds accepts 2 subkeys.

**Key expansion:**

- The 128-bit key is expanded into 52 16-bit keys,  $K_1, K_2, \dots, K_{52}$ .
- The key expansion is done differently for encryption than for decryption.
- Once the 52 keys are generated, the encryption and decryption operations are the same.
- The 52 encryption keys are generated by writing out the 128-bit key and, starting from the left, chopping off 16 bits at a time.
- This generates eight 16-bit keys



**One Round:**

- It has 17 rounds, where the odd numbered rounds are different from the even numbered rounds.
- Each round takes the input a 64-bit quantity and treats it as four 16-bit quantities  $X_a, X_b, X_c, X_d$ . Mathematical Operations are performed on it.
- In IDEA, Odd rounds accepts 4 subkeys.
- Even rounds accept 2 subkeys.

An odd round, therefore has as input  $X_a, X_b, X_c, X_d$  and keys  $K_a, K_b, K_c, K_d$ . An even round has as input  $X_a, X_b, X_c, X_d$  and keys  $K_e$  and  $K_f$ .

**Odd round:**

The odd round is simple.  $X_a$  is replaced by  $X_a \otimes K_a$ .  $X_d$  is replaced by  $X_d \otimes K_d$ .  $X_c$  is replaced by  $X_b + K_b$ .  $X_b$  is replaced by  $X_c + K_c$ .

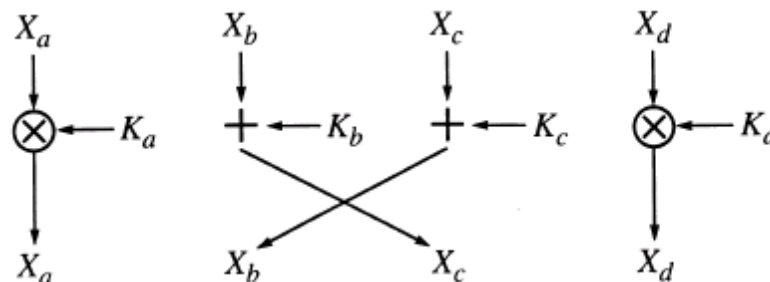
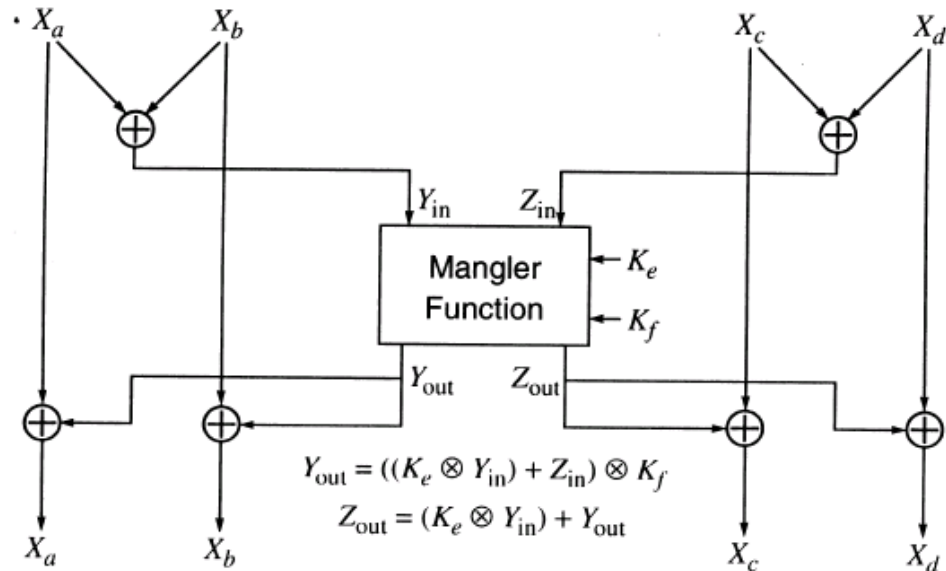


Figure: IDEA odd round

**Even Round:**

The even round is a little more complicated. Again, we have  $X_a$ ,  $X_b$ ,  $X_c$ , and  $X_d$ . We have two keys,  $K_e$  and  $K_f$ . We're going to first compute two values, which we'll call  $Y_{in}$  and  $Z_{in}$ . We'll do a function, which we'll call the *mangler function*, which takes as input  $Y_{in}$ ,  $Z_{in}$ ,  $K_e$ , and  $K_f$  and produces what we'll call  $Y_{out}$  and  $Z_{out}$ . We'll use  $Y_{out}$  and  $Z_{out}$  to modify  $X_a$ ,  $X_b$ ,  $X_c$ , and  $X_d$ .

$$Y_{in} = X_a \oplus X_b \quad Z_{in} = X_c \oplus X_d$$

$$Y_{out} = ((K_e \otimes Y_{in}) + Z_{in}) \otimes K_f$$

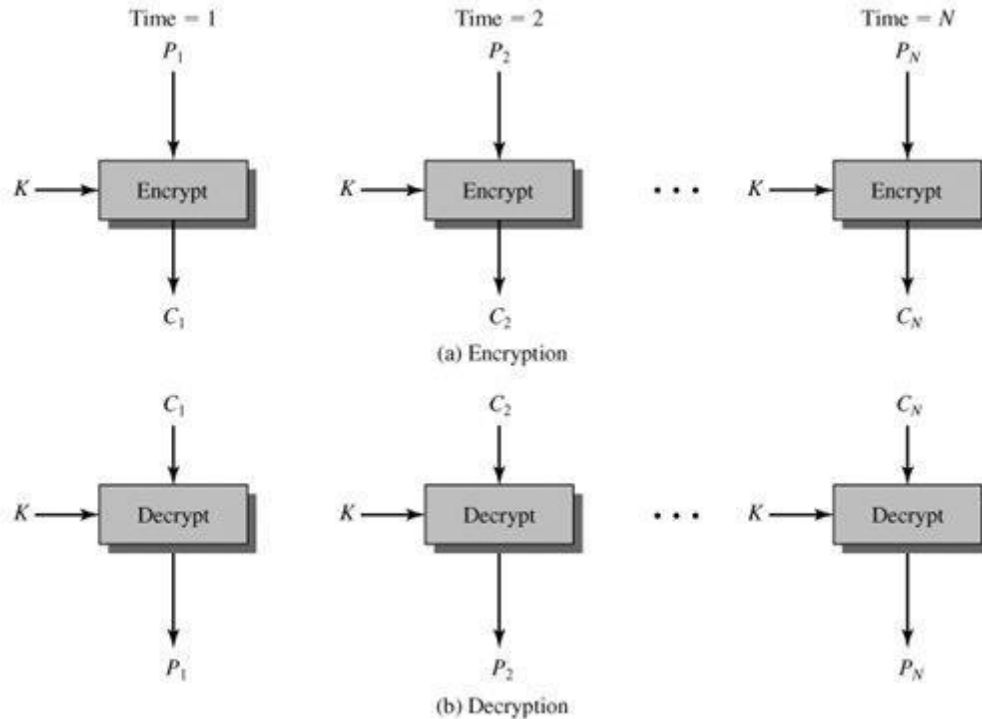
$$Z_{out} = (K_e \otimes Y_{in}) + Y_{out}$$

**BLOCK CIPHER MODES OF OPERATION:**

A block cipher algorithm is a basic building block for providing data security. To apply a block cipher in a variety of applications, different "modes of operation" have been defined by NIST. In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. The modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used.

**Electronic Codebook Mode:**

The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key (Figure a & b). The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext. For a message longer than b bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key. In Figure, the plaintext (padded as necessary) consists of a sequence of b-bit blocks,  $P_1, P_2, \dots, P_N$ ; the corresponding sequence of ciphertext blocks is  $C_1, C_2, \dots, C_N$ .



**Figure. Electronic Codebook (ECB) Mode**

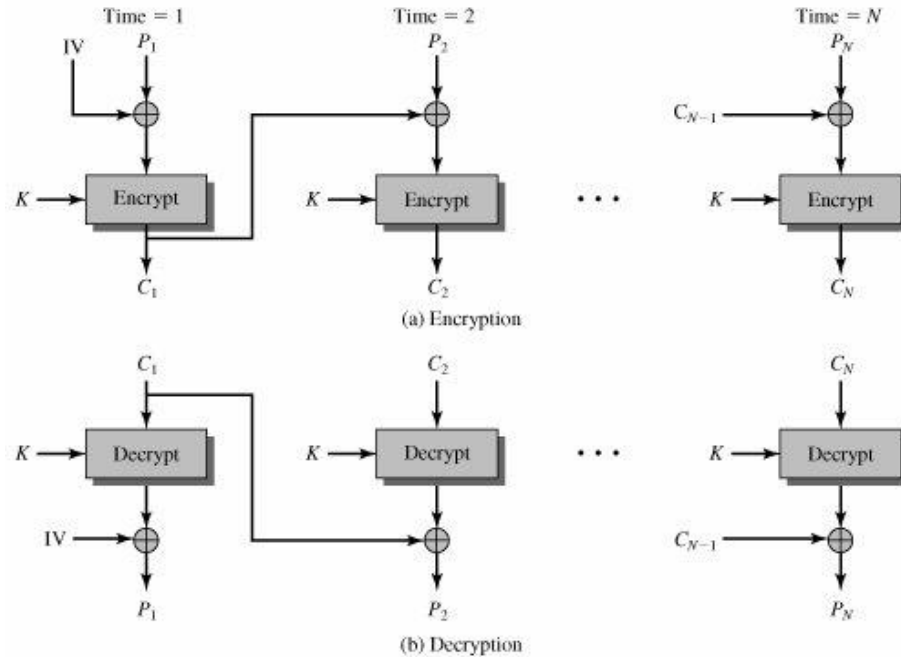
The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES key securely, ECB is the appropriate mode to use. The most significant characteristic of ECB is that the same  $b$ -bit block of plaintext, if it appears more than once in the message, always produces the same ciphertext.

For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with. If the message has repetitive elements, with a period of repetition a multiple of  $b$  bits, then these elements can be identified by the analyst. This may help in the analysis or may provide an opportunity for substituting or rearranging blocks.

### Cipher Block Chaining Mode:

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode.

In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of  $b$  bits are not exposed



**Figure : Cipher Block Chaining (CBC) Mode**

### Cipher Feedback Mode:

The DES scheme is essentially a block cipher technique that uses  $b$ -bit blocks. However, it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode. Figure depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is  $s$  bits; a common value is  $s = 8$ . As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than units of  $b$  bits, the plaintext is divided into segments of  $s$  bits.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit.

Let  $S_s(X)$  be defined as the most significant  $s$  bits of  $X$ . Then

$$C1 = P1 \oplus S_s[E(K, IV)]$$

Therefore,

$$P1 = C1 \oplus S_s [E(K, IV)]$$



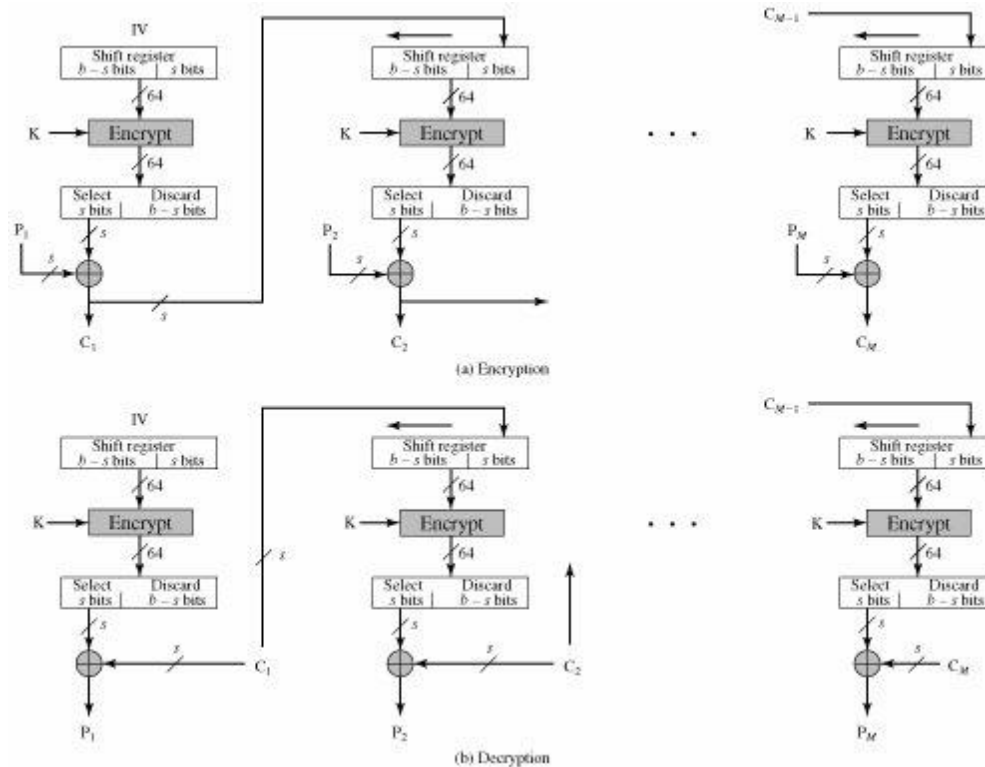


Figure. s-bit Cipher Feedback (CFB) Mode

### Counter Mode:

In CTR mode a counter, equal to the plaintext block size is used. The only requirement is that the counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block. For encryption, the counter is encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertext block to recover the corresponding plaintext block.

Advantages:

1. Hardware efficiency
2. Software efficiency
3. Preprocessing
4. Random access
5. Provable security
6. Simplicity

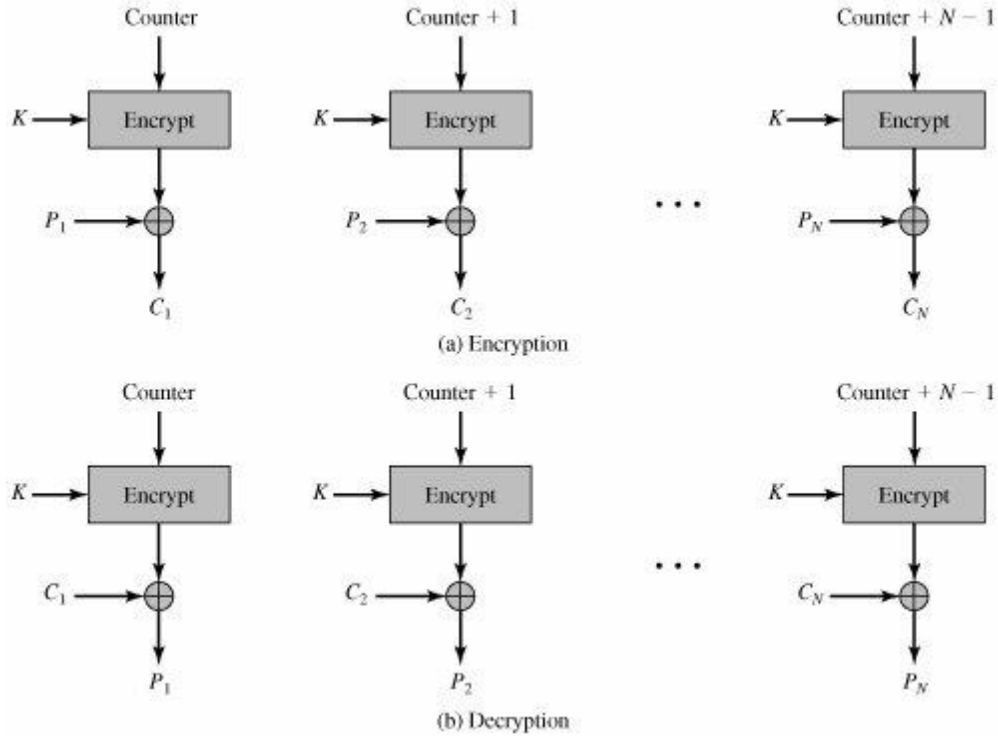


Figure. Counter (CTR) Mode



### UNIT- III

#### **PRIME NUMBER:**

Prime numbers only have divisors of 1 and self they cannot be written as a product of other numbers.

eg. 2,3,5,7 are prime, 4,6,8,9,10 are not

prime numbers are central to number theory

List of prime number less than 200 is:

2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131  
137 139 149 151 157 163 167 173 179 181 191 193 197 199

An integer  $p > 1$  is a prime number if and only if its only divisors are  $\pm 1$  and  $\pm p$ .

Any integer  $a > 1$  can be factored in a unique way as

where  $p_1 < p_2 < \dots < p_t$  are prime numbers and where each is a positive integer. This is known as the fundamental theorem of arithmetic

$$\begin{array}{ll} 91 & = 7 \times 13 \\ 3600 & = 2^4 \times 3^2 \times 5^2 \\ 11011 & = 7 \times 11^2 \times 13 \end{array}$$

If  $P$  is the set of all prime numbers, then any positive integer  $a$  can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \quad \text{where each } a_p \geq 0$$

The right-hand side is the product over all possible prime numbers  $p$ ; for any particular value of  $a$ , most of the exponents  $a_p$  will be 0.

#### **RELATIVELY PRIME NUMBERS:**

Two numbers  $a, b$  are relatively prime (coprime) if they have no common divisors apart from 1.

– eg. 8 and 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor.

#### **MODULAR ARITHMETIC:**

Given two positive integer  $n$  and  $a$ , if we divide  $a$  by  $n$ , we get an integer quotient  $q$  and an integer remainder  $r$  that obey the following relationship:

$$a = qn + r \quad 0 \leq r < n; q = \lfloor a/n \rfloor$$

#### **THE EUCLIDEAN ALGORITHM :**

One of the basic techniques of number theory is the Euclidean algorithm, which is a simple procedure for determining the **greatest common divisor** of two positive integers.

#### **Greatest Common Divisor:**

- **The greatest common divisor of  $a$  and  $b$  is the largest integer that divides both  $a$  and  $b$ .** We also define  $\text{gcd}(0, 0) = 0$ .
- The positive integer  $c$  is said to be the greatest common divisor of  $a$  and  $b$  if
  1.  $c$  is a divisor of  $a$  and of  $b$ ;



2. any divisor of a and b is a divisor of c.

- An equivalent definition is the following:  
 $\gcd(a, b) = \max\{k, \text{ such that } k|a \text{ and } k|b\}$   
 $\gcd(60, 24) = \gcd(60, -24) = 12$

In general,  $\gcd(a, b) = \gcd(|a|, |b|)$ .

### Finding the Greatest Common Divisor:

The Euclidean algorithm is based on the following theorem:

For any nonnegative integer a and any positive integer b,  $\gcd(a,b)=\gcd(b,a \bmod b)$

$\gcd(55, 22) = \gcd(22, 55 \bmod 22) = \gcd(22, 11) = 11$

$$\left. \begin{array}{ll} a = q_1 b + r_1 & 0 < r_1 < b \\ b = q_2 r_1 + r_2 & 0 < r_2 < r_1 \\ r_1 = q_3 r_2 + r_3 & 0 < r_3 < r_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ r_{n-2} = q_n r_{n-1} + r_n & 0 < r_n < r_{n-1} \\ r_{n-1} = q_{n+1} r_n + 0 & \\ d = \gcd(a, b) = r_n & \end{array} \right\}$$

#### Example GCD(1970,1066)

$$1970 = 1 \times 1066 + 904 \quad \gcd(1066, 904)$$

$$1066 = 1 \times 904 + 162 \quad \gcd(904, 162)$$

$$904 = 5 \times 162 + 94 \quad \gcd(162, 94)$$

$$162 = 1 \times 94 + 68 \quad \gcd(94, 68)$$

$$94 = 1 \times 68 + 26 \quad \gcd(68, 26)$$

$$68 = 2 \times 26 + 16 \quad \gcd(26, 16)$$

$$26 = 1 \times 16 + 10 \quad \gcd(16, 10)$$

$$16 = 1 \times 10 + 6 \quad \gcd(10, 6)$$

$$10 = 1 \times 6 + 4 \quad \gcd(6, 4)$$

$$6 = 1 \times 4 + 2 \quad \gcd(4, 2)$$

$$4 = 2 \times 2 + 0 \quad \gcd(2, 0)$$

$$\gcd(1970, 1066) = 2$$

### CONGRUENT MODULO:

Two integers a and b are said to be congruent modulo of n if

$$a \bmod n = b \bmod n.$$

then this is written as  $a \equiv b \pmod n$ .

Ex: a=73 b=4 and n=23

$$73 \bmod 23 = 4$$

$$4 \bmod 23 = 4$$

So  $73 \equiv 4 \pmod{23}$



### Properties of Congruences:

Congruences have the following properties:

1.  $a \equiv b \pmod{n}$  if  $n|(a - b)$ .
2.  $a \equiv b \pmod{n}$  implies  $b \equiv a \pmod{n}$ .
3.  $a \equiv b \pmod{n}$  and  $b \equiv c \pmod{n}$  imply  $a \equiv c \pmod{n}$ .

### **Modular Arithmetic Operations:**

Many complex cryptographic algorithms are actually based on simple arithmetic. In modular arithmetic the numbers which going to deal are just integers and operations are addition, subtraction, multiplication and division.

Modular arithmetic exhibits the following properties:

1.  $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2.  $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3.  $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

## **FERMAT'S AND EULER'S THEOREMS**

### **Fermat's Theorem:**

Fermat's theorem states the following: If 'p' is prime and 'a' is a positive integer not divisible by p, then

$$a^{p-1} \equiv 1 \pmod{p}$$

*Proof:* Consider the set of positive integers less than  $p$ :  $\{1, 2, \dots, p - 1\}$  and multiply each element by  $a$ , modulo  $p$ , to get the set  $X = \{a \bmod p, 2a \bmod p, \dots, (p - 1)a \bmod p\}$ . None of the elements of  $X$  is equal to zero because  $p$  does not divide  $a$ . Furthermore, no two of the integers in  $X$  are equal. To see this, assume that  $ja \equiv ka \pmod{p}$ , where  $1 \leq j < k \leq p - 1$ . Because  $a$  is relatively prime<sup>5</sup> to  $p$ , we can eliminate  $a$  from both sides of the equation [see Equation (4.3)] resulting in  $j \equiv k \pmod{p}$ . This last equality is impossible, because  $j$  and  $k$  are both positive integers less than  $p$ . Therefore, we know that the  $(p - 1)$  elements of  $X$  are all positive integers with no two elements equal. We can conclude the  $X$  consists of the set of integers  $\{1, 2, \dots, p - 1\}$  in some order. Multiplying the numbers in both sets ( $p$  and  $X$ ) and taking the result mod  $p$  yields

$$a \times 2a \times \dots \times (p - 1)a \equiv [(1 \times 2 \times \dots \times (p - 1)) \pmod{p}]$$

$$a^{p-1}(p - 1)! \equiv (p - 1)! \pmod{p}$$

We can cancel the  $(p-1)!$  term because it is relatively prime to  $p$  then

$$a^{p-1} \equiv 1 \pmod{p}$$

**Euler's Totient Function:**

- It is defined as the number of positive integers less than 'n' and relatively prime to 'n' and is written as  $\phi(n)$ . By convention  $\phi(1)=1$ .
- It should be clear that, for a prime number p,

$$\phi(p) = p - 1$$

$$\phi(37) = 36$$

- To determine  $\phi(35)$ , we list all of the positive integers less than 35 that are relatively prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34

There are 24 numbers on the list, so  $\phi(35) = 24$

Now suppose that we have two prime numbers  $p$  and  $q$  with  $p \neq q$ . Then we can show that, for  $n = pq$ ,

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p - 1) \times (q - 1)$$

$$\phi(21) = \phi(3) \times \phi(7) = (3 - 1) \times (7 - 1) = 2 \times 6 = 12$$

where the 12 integers are {1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20}.

**Euler's Theorem:**

Euler's theorem states that for every  $a$  and  $n$  that are relatively prime:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

*Proof:* Equation (8.4) is true if  $n$  is prime, because in that case,  $\phi(n) = (n - 1)$  and Fermat's theorem holds. However, it also holds for any integer  $n$ . Recall that  $\phi(n)$  is the number of positive integers less than  $n$  that are relatively prime to  $n$ . Consider the set of such integers, labeled as

$$R = \{x_1, x_2, \dots, x_{\phi(n)}\}$$

That is, each element  $x_i$  of  $R$  is a unique positive integer less than  $n$  with  $\gcd(x_i, n) = 1$ . Now multiply each element by  $a$ , modulo  $n$ :

$$S = \{(ax_1 \pmod{n}), (ax_2 \pmod{n}), \dots, (ax_{\phi(n)} \pmod{n})\}$$

The set  $S$  is a permutation<sup>6</sup> of  $R$ , by the following line of reasoning:

1. Because  $a$  is relatively prime to  $n$  and  $x_i$  is relatively prime to  $n$ ,  $ax_i$  must also be relatively prime to  $n$ . Thus, all the members of  $S$  are integers that are less than  $n$  and that are relatively prime to  $n$ .



2. There are no duplicates in  $S$ . Refer to Equation (4.5). If  $ax_i \bmod n = ax_j \bmod n$ , then  $x_i = x_j$ .

Therefore,

$$\begin{aligned} \prod_{i=1}^{\phi(n)} (ax_i \bmod n) &= \prod_{i=1}^{\phi(n)} x_i \\ \prod_{i=1}^{\phi(n)} ax_i &\equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \\ a^{\phi(n)} \times \left[ \prod_{i=1}^{\phi(n)} x_i \right] &\equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n} \\ a^{\phi(n)} &\equiv 1 \pmod{n} \end{aligned}$$

which completes the proof. This is the same line of reasoning applied to the proof of Fermat's theorem.

### **THE CHINESE REMAINDER THEOREM:**

The **Chinese remainder theorem** (CRT) is used to solve a set of congruent equations with one variable but different moduli, which are relatively prime, as shown below:

$$\begin{aligned} x &\equiv a_1 \pmod{m_1} \\ x &\equiv a_2 \pmod{m_2} \\ &\dots \\ x &\equiv a_k \pmod{m_k} \end{aligned}$$

The Chinese remainder theorem states that the above equations have a unique solution if the moduli are relatively prime.

**SOLUTION** The solution to the set of equations follows these steps:

1. Find  $M = m_1 \times m_2 \times \dots \times m_k$ . This is the common modulus.
2. Find  $M_1 = M/m_1, M_2 = M/m_2, \dots, M_k = M/m_k$ .
3. Find the multiplicative inverse of  $M_1, M_2, \dots, M_k$  using the corresponding moduli  $(m_1, m_2, \dots, m_k)$ . Call the inverses



$$M_1^{-1}, M_2^{-1}, \dots, M_k^{-1}.$$

4. The solution to the simultaneous equations is

$$x = (a_1 \times M_1 \times M_1^{-1} + a_2 \times M_2 \times M_2^{-1} + \dots + a_k \times M_k \times M_k^{-1}) \bmod M$$

Example:

$$x \equiv 2 \pmod{3}$$

$$x \equiv 3 \pmod{5}$$

$$x \equiv 2 \pmod{7}$$

1.  $M = 3 \times 5 \times 7 = 105$
2.  $M_1 = 105/3 = 35, M_2 = 105/5 = 21, M_3 = 105/7 = 15$
3. The inverses are  $M_1^{-1} = 2, M_2^{-1} = 1, M_3^{-1} = 1$
4.  $x = (2 \times 35 \times 2 + 3 \times 21 \times 1 + 2 \times 15 \times 1) \bmod 105 = 23 \bmod 105$

### DISCRETE LOGARITHMS:

#### Discrete Logarithms

- > From Euler's theorem that, **a** and **n** which are relatively prime, We have,  $a^{\phi(n)} \equiv 1 \pmod{n}$  where  $\phi(n)$  is Euler's totient function that gives the number of positive integers less than **n** and relatively prime to **n**
- > Let  $a^m = 1 \pmod{n}$
- > If **a** and **n** are relatively prime, then there is at least one integer **m** that satisfies the given equation
- > Least positive exponent **m** for which the above equation holds is referred to in several ways:
  - o Order of **a** (mod n)
  - o Exponent to which **a** belongs (mod n)
  - o Length of the period generated by **a**

$7_1$		$7 \pmod{19}$
$7_2$	$= 49 = 2 \times 19 + 11$	$11 \pmod{19}$
$7_3$	$= 343 = 18 \times 19 + 1$	$1 \pmod{19}$
$7_4$	$= 2401 = 126 \times 19 + 7$	$7 \pmod{19}$
$7_5$	$= 16807 = 884 \times 19 + 11$	$11 \pmod{19}$





$a$	$a^2$	$a^3$	$a^4$	$a^5$	$a^6$	$a^7$	$a^8$	$a^9$	$a^{10}$	$a^{11}$	$a^{12}$	$a^{13}$	$a^{14}$	$a^{15}$	$a^{16}$	$a^{17}$	$a^{18}$
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

*Table. Powers of Integers, Modulo 19*

### Logarithms for Modular Arithmetic

- The Logarithm function is the inverse of exponentiation for ordinary real numbers
- The Logarithm of a number is defined to be the power to which some positive base (except 1) must be raised in order to equal the number
- That is, for base  $x$  and for a value  $y$ ,  $y = x^{\log_x(y)}$

Properties of logarithms:

$$\log_x(1) = 0$$

$$\log_x(x) = 1$$

$$\log_x(yz) = \log_x(y) + \log_x(z)$$

$$\log_x(y^r) = r \times \log_x(y)$$

## **PUBLIC KEY CRYPTOGRAPHY:**

### **Introduction:**

- Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys - one a public key and one a private key. It is also known as public-key encryption.
- Asymmetric encryption transforms plaintext into ciphertext using a one of two keys and an encryption algorithm. Using the paired key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- Asymmetric encryption can be used for confidentiality, authentication, or both.
- The most widely used public-key cryptosystem is RSA.

### **Principles of Public-Key Cryptosystems:**

The concept of public key cryptography is invented for two most difficult problems of Symmetric key encryption.



- **key distribution** – how to have secure communications in general without having to trust a KDC (key distribution center) with your key.
- **digital signatures** – how to verify a message comes intact from the claimed sender.

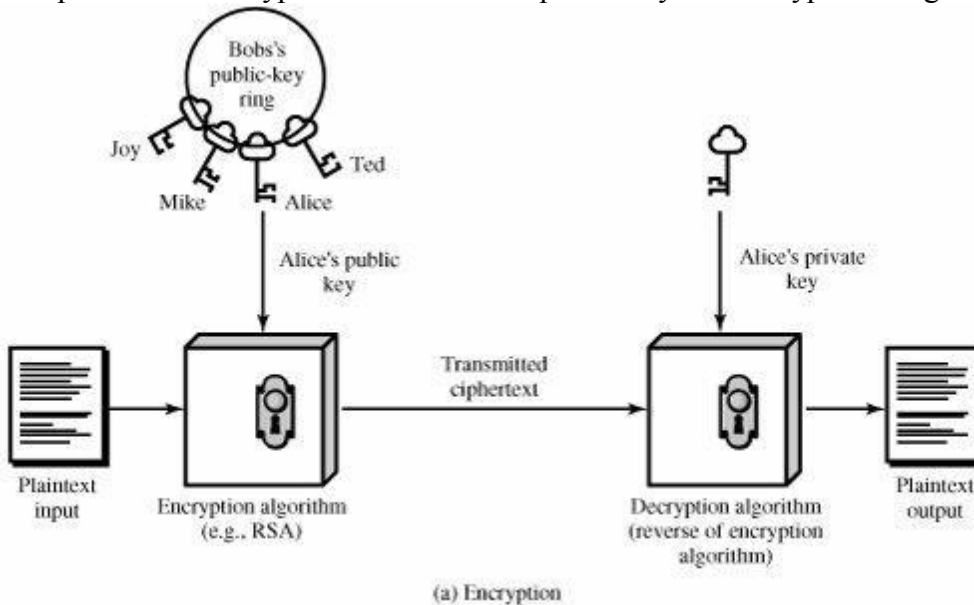
### Public-Key Cryptosystems:

A public-key encryption scheme has six ingredients

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

### ENCRYPTION:

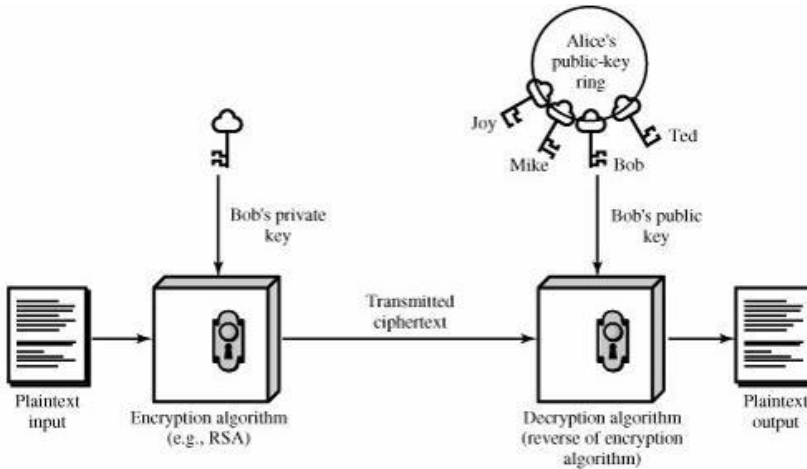
The plaintext is encrypted with receiver's public key and decrypted using receiver private key.



### AUTHENTICATION:

Plaintext is encrypted with sender's private key and decrypted using sender's public key.

- The act of messages ciphertext getting decrypted by sender's public key is the proof that the message is actually sent by the designated sender.



(b) Authentication

### Difference between symmetric and public key encryption:

Conventional Encryption	Public-Key Encryption
<p><i>Needed to Work:</i></p> <ol style="list-style-type: none"> <li>1. The same algorithm with the same key is used for encryption and decryption.</li> <li>2. The sender and receiver must share the algorithm and the key.</li> </ol> <p><i>Needed for Security:</i></p> <ol style="list-style-type: none"> <li>1. The key must be kept secret.</li> <li>2. It must be impossible or at least impractical to decipher a message if no other information is available.</li> <li>3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.</li> </ol>	<p><i>Needed to Work:</i></p> <ol style="list-style-type: none"> <li>1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption.</li> <li>2. The sender and receiver must each have one of the matched pair of keys (not the same one).</li> </ol> <p><i>Needed for Security:</i></p> <ol style="list-style-type: none"> <li>1. One of the two keys must be kept secret.</li> <li>2. It must be impossible or at least impractical to decipher a message if no other information is available.</li> <li>3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.</li> </ol>

- Examples for conventional encryption are DES, AES, IDEA and Blowfish.
- Examples for public key encryption are RSA, Diffie-Hellman, Elliptic Curve Cryptography.

There is some source A that produces a message in plaintext,  $X = [X_1, X_2, \dots, X_M]$ . The M elements of X are letters in some finite alphabet. The message is intended for destination B. B generates a related pair of keys: a public key, PUB, and a private key, PRb. PRb is known only to B, whereas PUB is publicly available and therefore accessible by A.

With the message X and the encryption key PUB as input, A forms the ciphertext  $Y = [Y_1, Y_2, \dots, Y_N]$ :

$$Y = E(\text{PUB}, X)$$

The intended receiver, in possession of the matching private key, is able to invert the transformation:

$$X = D(\text{PRb}, Y)$$

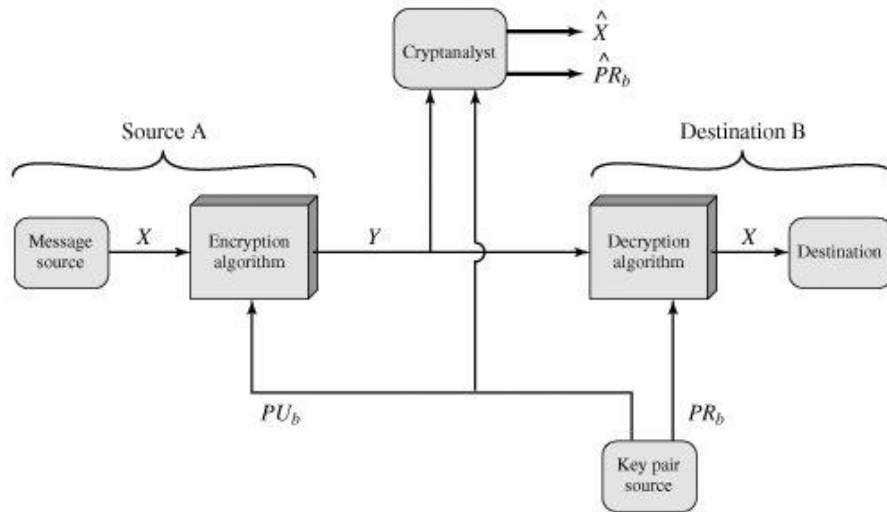


Figure: public key cryptosystems: Secrecy (or) confidentiality

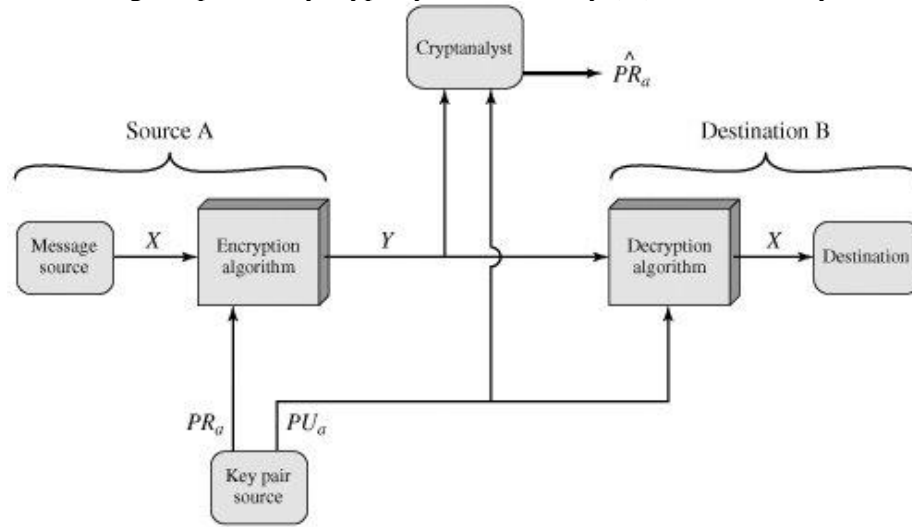


Figure: Public-Key Cryptosystem: Authentication

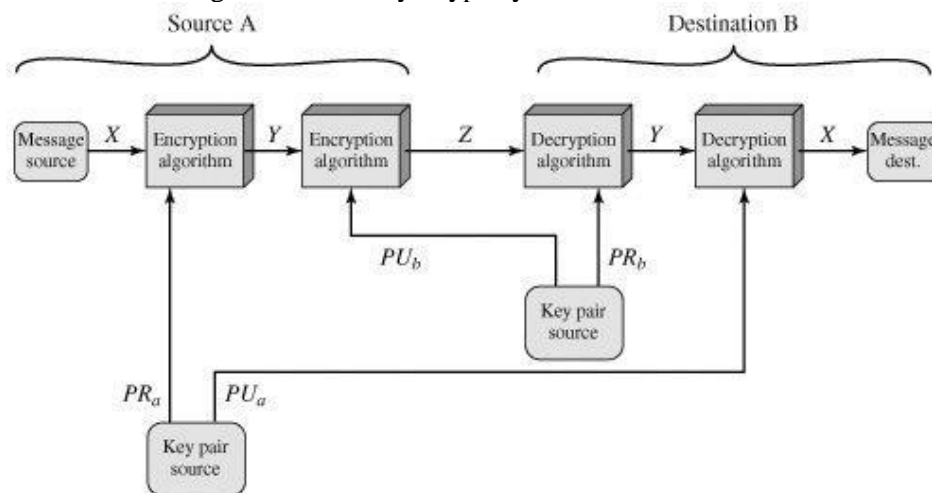


Figure: Public-Key Cryptosystem: Authentication and Secrecy



### 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.
- **Key exchange:** Two sides cooperate to exchange a session key.

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

### Requirements for Public-Key Cryptography:

1. It is computationally easy for a party B to generate a pair (public key PUB, private key PRb).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:  $C = E(\text{PUB}, M)$
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:  $M = D(\text{PRb}, C) = D[\text{PRb}, E(\text{PUB}, M)]$
4. It is computationally infeasible for an adversary, knowing the public key, PUB, to determine the private key, PRb.
5. It is computationally infeasible for an adversary, knowing the public key, PUB, and a ciphertext, C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

The two keys can be applied in either order:

$$M = D[\text{PUB}, E(\text{PRb}, M)] = D[\text{PRb}, E(\text{PUB}, M)]$$

### THE RSA ALGORITHM

RSA is a public key encryption algorithm developed by Rivest(R) , Shamir(S) and Adleman (A) in year 1977. The RSA scheme is a block cipher in which the plaintext & ciphertext are integers between 0 and n-1 for some 'n'. A typical size for 'n' is 1024 bits or 309 decimal digits. RSA algorithm uses an expression with exponentials.

- In RSA 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  $\log_2(n)+1$
- RSA uses two exponents 'e' and 'd' where e  $\rightarrow$  public and d  $\rightarrow$  private.
- Encryption and decryption are of following form, for some PlainText 'M' and CipherText block 'C'

$$C = M^e \text{ mod } n$$

$$M = C^d \text{ mod } n$$

$$M = C^d \text{ mod } n = (M^e \text{ mod } n)^d \text{ mod } n = (M^e)^d \text{ mod } n = M^{ed} \text{ mod } n$$



- Both sender and receiver must know the value of  $n$ .
- The sender knows the value of 'e' & only the receiver knows the value of 'd' thus this is a public key encryption algorithm with a  
Public key  $PU = \{e, n\}$   
Private key  $PR = \{d, n\}$

**Requirements:**

The RSA algorithm to be satisfactory for public key encryption, the following requirements must be met:

- It is possible to find values of  $e$ ,  $d$  and  $n$  such that “ $Med \bmod n = M$ ” for all  $M < n$
- It is relatively easy to calculate “ $Me \bmod n$ ” and “ $Cd \bmod n$ ” for all values of  $M < n$
- It is infeasible to determine “ $d$ ” given ‘e’ & ‘n’.

**Key Generation**

Select $p, q$	$p$ and $q$ both prime, $p \neq q$
Calculate $n = p \times q$	
Calculate $\phi(n) = (p - 1)(q - 1)$	
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**

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

**Decryption**

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



For decryption, we calculate  $M = 11^{23} \bmod 187$ :

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

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

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

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

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

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

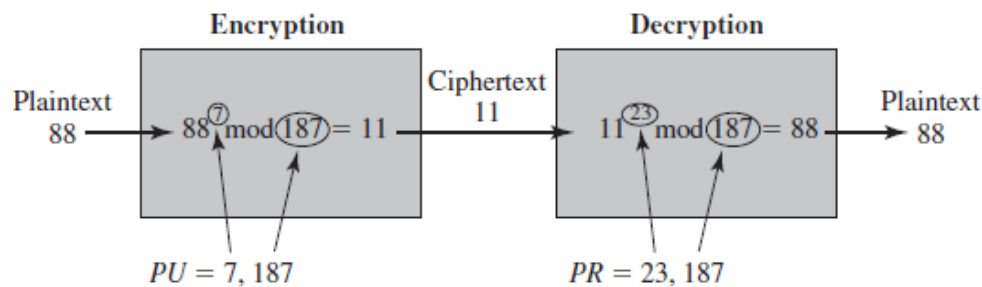


Figure 9.6 Example of RSA Algorithm

### Diffie-Hellman key exchange/Agreement Algorithm:

Whitefield Diffie and Martin Hellman devised an amazing solution to the problem of key agreement, or key exchange, in 1976. This solution is called the Diffie—Hellman key exchange/agreement algorithm. The beauty of this scheme is that the two parties, who want to communicate securely, can agree on a symmetric key using this technique. This key can then be used for encryption/decryption. However, we must note that the Diffie-Hellman key exchange algorithm can be used only for key agreement, but not for encryption or decryption of messages. Once both the parties agree on the key to be used, they need to use other symmetric key-encryption algorithms for actual encryption or decryption of messages. Although the Diffie—Hellman key-exchange algorithm is based on mathematical principles, it is quite simple to understand.

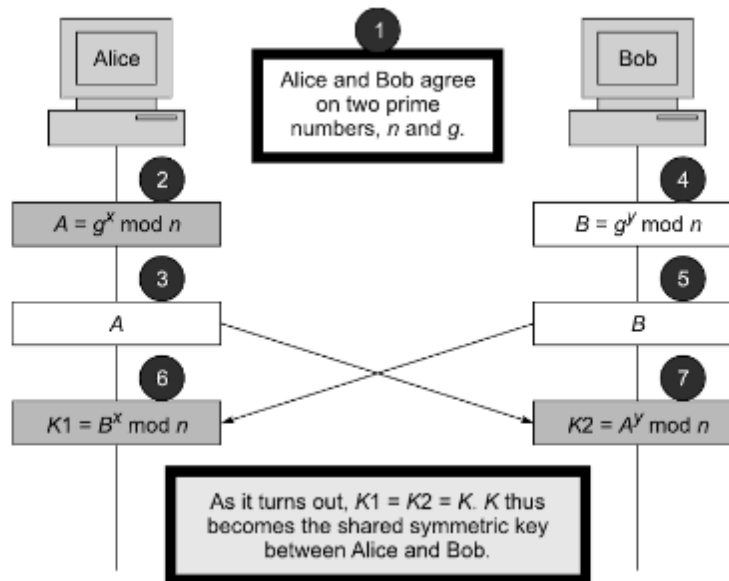
Description of the algorithm:



Let us assume that Alice and Bob want to agree upon a key to be used for encrypting/decrypting messages that would be exchanged between them. Then, the Diffie–Hellman key-exchange algorithm works as shown in Fig. 2.52.

1. Firstly, Alice and Bob agree on two large prime numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number  $x$ , and calculates  $A$  such that:  
 $A = g^x \text{ mod } n$
3. Alice sends the number  $A$  to Bob.
4. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that:  
 $B = g^y \text{ mod } n$
5. Bob sends the number  $B$  to Alice.
6. A now computes the secret key  $K1$  as follows:  
 $K1 = B^x \text{ mod } n$
7. B now computes the secret key  $K2$  as follows:  
 $K2 = A^y \text{ mod } n$

**Fig. 2.52** Diffie-Hellman key-exchange algorithm



**Example of the algorithm:**





1. Firstly, Alice and Bob agree on two large prime numbers,  $n$  and  $g$ . These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.

Let  $n = 11, g = 7$ .

2. Alice chooses another large random number  $x$ , and calculates  $A$  such that:  
 $A = g^x \text{ mod } n$

Let  $x = 3$ . Then, we have,  $A = 7^3 \text{ mod } 11 = 343 \text{ mod } 11 = 2$ .

3. Alice sends the number  $A$  to Bob.

Alice sends 2 to Bob.

4. Bob independently chooses another large random integer  $y$  and calculates  $B$  such that  
 $B = g^y \text{ mod } n$

Let  $y = 6$ . Then, we have,  $B = 7^6 \text{ mod } 11 = 117649 \text{ mod } 11 = 4$ .

5. Bob sends the number  $B$  to Alice.

Bob sends 4 to Alice.

6. A now computes the secret key  $K1$  as follows:  
 $K1 = B^x \text{ mod } n$

We have,  $K1 = 4^3 \text{ mod } 11 = 64 \text{ mod } 11 = 9$ .

7. B now computes the secret key  $K2$  as follows:  
 $K2 = A^y \text{ mod } n$

We have,  $K2 = 2^6 \text{ mod } 11 = 64 \text{ mod } 11 = 9$ .

### PROBLEM WITH THE ALGORITHM (MAN-IN-THE-MIDDLE ATTACK):

Can we now consider that the Diffie—Hellman key-exchange algorithm solves all our problems associated with key exchange? Unfortunately, not quite! The Diffie-Hellman key exchange algorithm can fall pray to the man-in-the-middle attack

ST.ANN'S COLLEGE OF ENGINEERING & TECHNOLOGY,CHIRALA  
CSE SACET CSE

1. Alice wants to communicate with Bob securely, and therefore, she first wants to do a Diffie-Hellman key exchange with him. For this purpose, she sends the values of  $n$  and  $g$  to Bob, as usual. Let  $n = 11$  and  $g = 7$ . (As usual, these values Will form the basis of Alice's  $A$  and Bob's  $B$ , which will be used to calculate the symmetric key  $K1 = K2 = K$ )

2. Alice does not realize that the attacker Tom is listening quietly to the conversation between her and Bob. Tom simply picks up the values of  $n$  and  $g$ , and also forwards them to Bob as they originally were (i.e.  $n = 11$  and  $g = 7$ ).

Alice	Tom	Bob
$n = 11, g = 7$	$n = 11, g = 7$	$n = 11, g = 7$

3. Now, let us assume that Alice, Tom and Bob select random numbers  $x$  and  $y$  as shown in Figure.

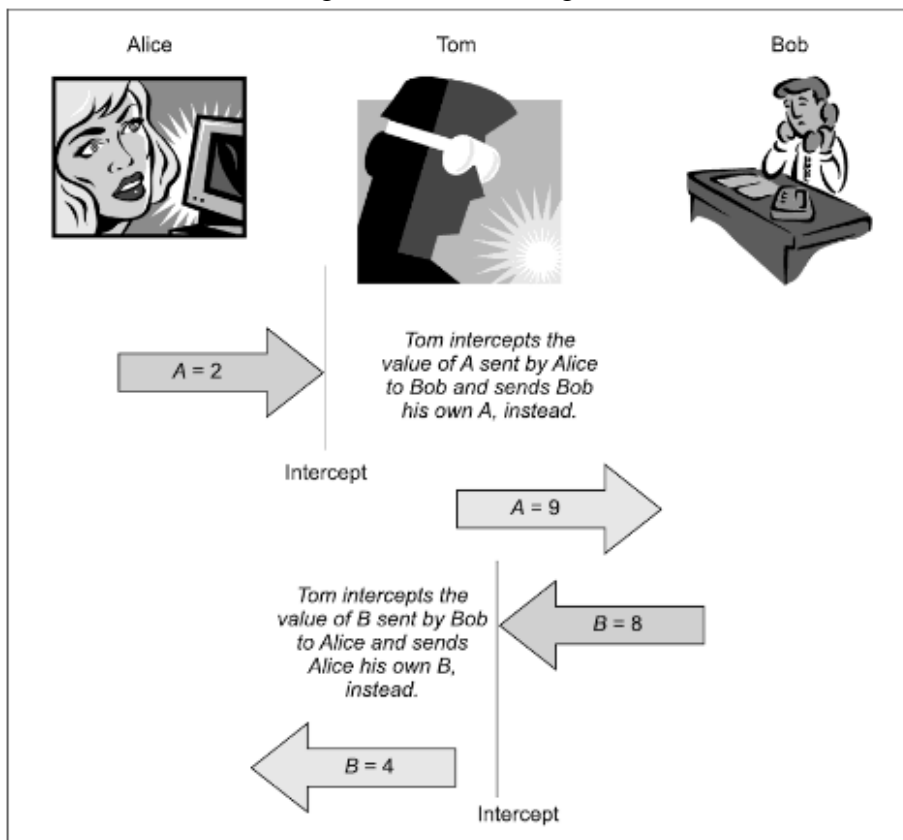
Alice	Tom	Bob
$x = 3$	$x = 8, y = 6$	$y = 9$



4. One question at this Stage could be: Why does Tom select both  $x$  and  $y$ ? We shall answer that shortly. Now, based on these values, all the three persons calculate the values of  $A$  and  $B$  as shown in Figure, note that Alice and Bob calculate only  $A$  and  $B$ , respectively. However, Tom calculates both  $A$  and  $B$ .

Alice	Tom	Bob
$A = g^x \text{ mod } n$	$A = g^x \text{ mod } n$	$B = g^y \text{ mod } n$
$= 7^3 \text{ mod } 11$	$= 7^8 \text{ mod } 11$	$= 7^9 \text{ mod } 11$
$= 343 \text{ mod } 11$	$= 5764801 \text{ mod } 11$	$= 40353607 \text{ mod } 11$
$= 2$	$= 9$	$= 8$
	$B = g^y \text{ mod } n$	
	$= 7^6 \text{ mod } 11$	
	$= 117649 \text{ mod } 11$	
	$= 4$	

5. Now the real drama begins as shown in figure,



As shown in the figure, the following things happen:

- Alice sends her (i.e. 2) to Bob. Tom intercepts it, and instead, sends his  $A$  (i.e. 9) to Bob. Bob has no idea that Tom had hijacked Alice's  $A$  and has instead given his  $A$  to Bob.
- In return, Bob sends his  $B$  (i.e. 8) to Alice. As before, Tom intercepts it, and instead, sends his  $B$  (i.e. 4) to Alice. Alice thinks that this  $B$  came from Bob. She has no idea that Tom had intercepted the transmission from Bob, and changed  $B$ .
- Therefore, at this juncture, Alice, Tom and Bob have the values Of  $A$  and  $B$  as shown in following Fig.



Alice	Tom	Bob
$A = 2, B = 4^*$	$A = 2, B = 8$	$A = 9^*, B = 8$
(Note: * indicates that these are the values after Tom hijacked and changed them.)		

Based on these values, all The three persons now calculate their keys as shown in following Figure. We will notice that Alice calculates only K1 , Bob calculates only K2, whereas Tom calculates both K1 and K2. Why does Tom need to do this?

Alice	Tom	Bob
$K1 = B^x \text{ mod } n$	$K1 = B^x \text{ mod } n$	$K2 = A^y \text{ mod } n$
$= 4^3 \text{ mod } 11$	$= 8^8 \text{ mod } 11$	$= 9^9 \text{ mod } 11$
$= 64 \text{ mod } 11$	$= 16777216 \text{ mod } 11$	$= 387420489 \text{ mod } 11$
$= 9$	$= 5$	$= 5$
	$K2 = A^y \text{ mod } n$	
	$= 2^6 \text{ mod } 11$	
	$= 64 \text{ mod } 11$	
	$= 9$	

### ELGAMAL CRYPTOGRAPHY:

Taber ElGamal created ElGamal cryptography, more popularly known as ElGamal cryptosystem. There are three aspects that need to be discussed: ElGamal key generation, ElGamal encryption, and ElGamal decryption.

#### Elgamal key generation:

This involves the following steps:

1. Select a large prime number called  $P$ . This is the first part of the encryption key or public key.
2. Select the decryption key or private key  $D$ . There are some mathematical rules that need to be followed here, which we are omitting for keeping things simple.
3. Select the second part of the encryption key or public key  $E1$ .
4. The third part of the encryption key or public key  $E2$  is computed as  $E2 = E1^D \text{ mod } P$ .
5. The public key is  $(E1, E2, P)$  and the private key is  $D$ .

For example,  $P = 11, E1 = 2, D = 3$ . Then  $E2 = E1^D \text{ mod } P = 2^3 \text{ mod } 11 = 8$ .

Hence, the public key is  $(2, 8, 11)$  and the private key is  $3$ .



### Elgamal key encryption:

This involves the following steps:

1. Select a random integer  $R$  that fulfills some mathematical properties, which are ignored here.
2. Compute the first part of the cipher text  $C1 = E1^R \text{ mod } P$ .
3. Compute the second part of the cipher text  $C2 = (PT \times E2^R) \text{ mod } P$ , where  $PT$  is the plain text.
4. The final cipher text is  $(C1, C2)$ .

In our example, let  $R = 4$  and plain text  $PT = 7$ . Then we have:

$$C1 = E1^R \text{ mod } P = 2^4 \text{ mod } 11 = 16 \text{ mod } 11 = 5$$

$$C2 = (PT \times E2^R) \text{ mod } P = (7 \times 2^8) \text{ mod } 11 = (7 \times 4096) \text{ mod } 11 = 6$$

Hence, our cipher text is  $(5, 6)$ .

### Elgamal key decryption:

This involves the following step:

Compute the plain text  $PT$  using the formula  $PT = [C2 \times (C1^D)^{-1}] \text{ mod } P$

In our example:

$$PT = [C2 \times (C1^D)^{-1}] \text{ mod } P$$

$$PT = [6 \times (5^3)^{-1}] \text{ mod } 11 = [6 \times 3] \text{ mod } 11 = 7, \text{ which was our original plain text.}$$

### ELLIPTIC CURVE CRYPTOGRAPHY(ECC):

Elliptic Curve Cryptography (ECC) was discovered in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington) as an alternative mechanism for implementing public-key cryptography. Public-key algorithms create a mechanism for sharing keys among large numbers of participants or entities in a complex information system. Unlike other popular algorithms such as RSA, ECC is based on discrete logarithms that is much more difficult to challenge at equivalent key lengths.

**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**

$$K = n_B \times P_A$$

## UNIT-IV

### 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).

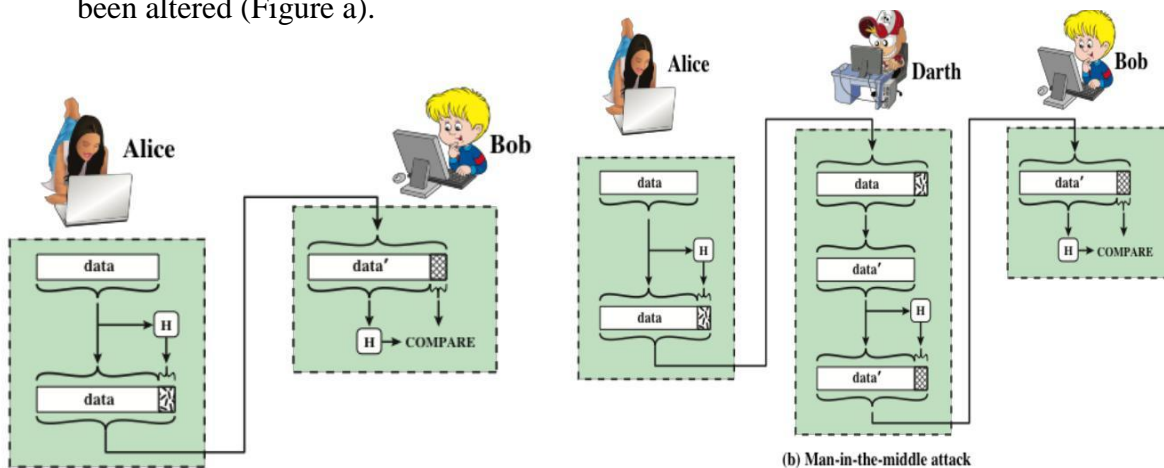


Figure 11.2 Attack Against Hash Function

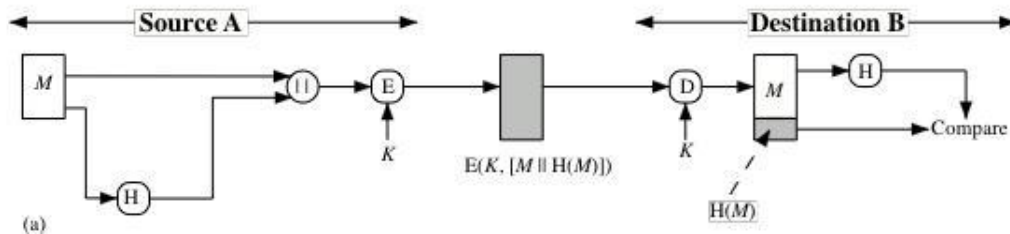


The hash function must be transmitted in a secure fashion. That is, the hash function must be protected so that if an adversary alters or replaces the message, it is not feasible for adversary to also alter the hash value to fool the receiver. This type of attack is shown in Figure b. In this example, 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.

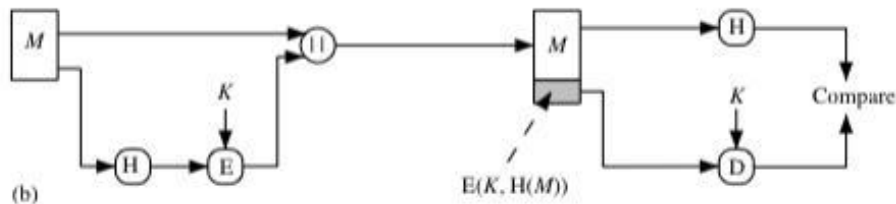
### Ways Hash Code Can Be Used to Provide Message Authentication:

A variety of ways in which a hash code can be used to provide message authentication, as follows:

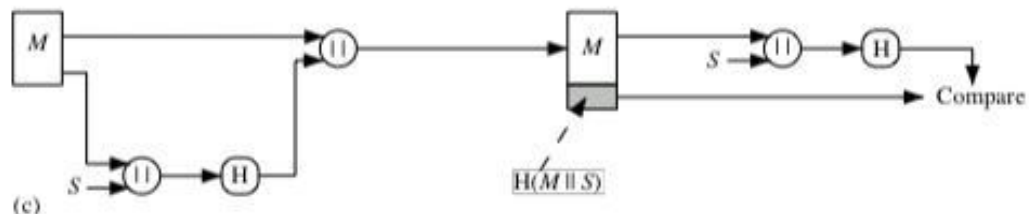
a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied 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.

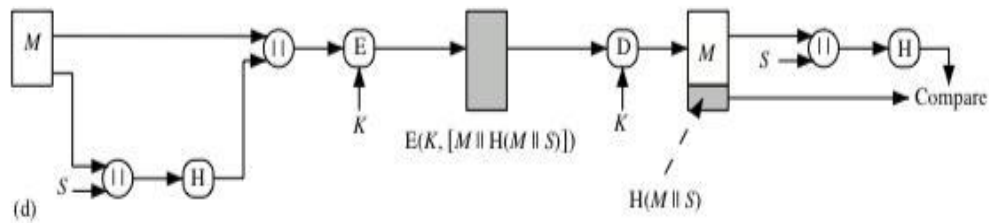


c) It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value  $S$ . A computes the hash value over the concatenation of  $M$  and  $S$  and appends the resulting hash value to  $M$ . Because 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.

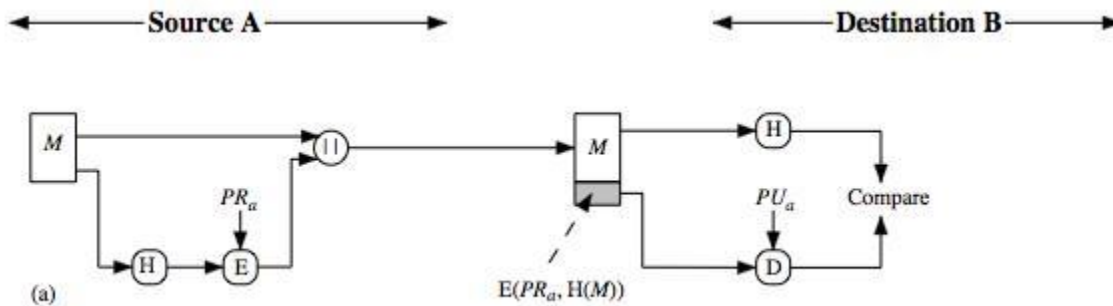


### Digital Signatures:

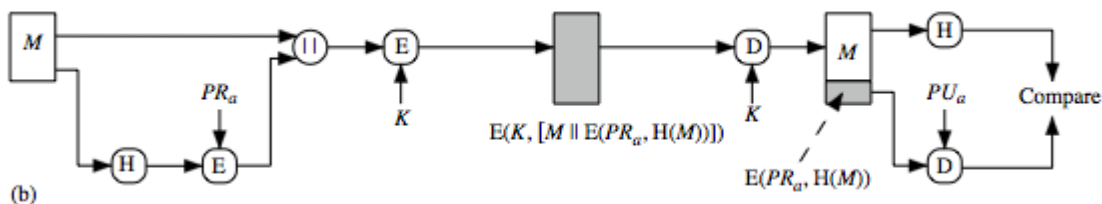
In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

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  $x \neq y$  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

### 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.

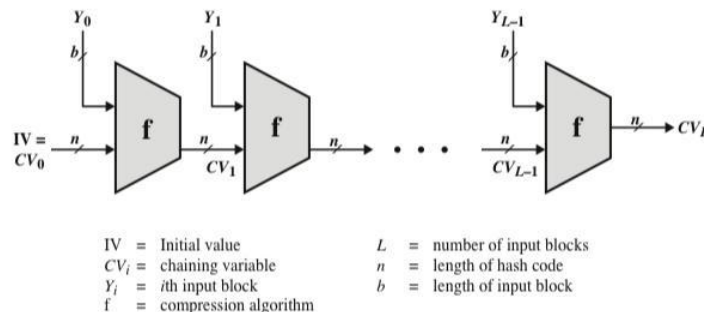


Figure 11.8 General Structure of Secure Hash Code

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.

### **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 2128 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 =  $896 \pmod{1024}$ ]. 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

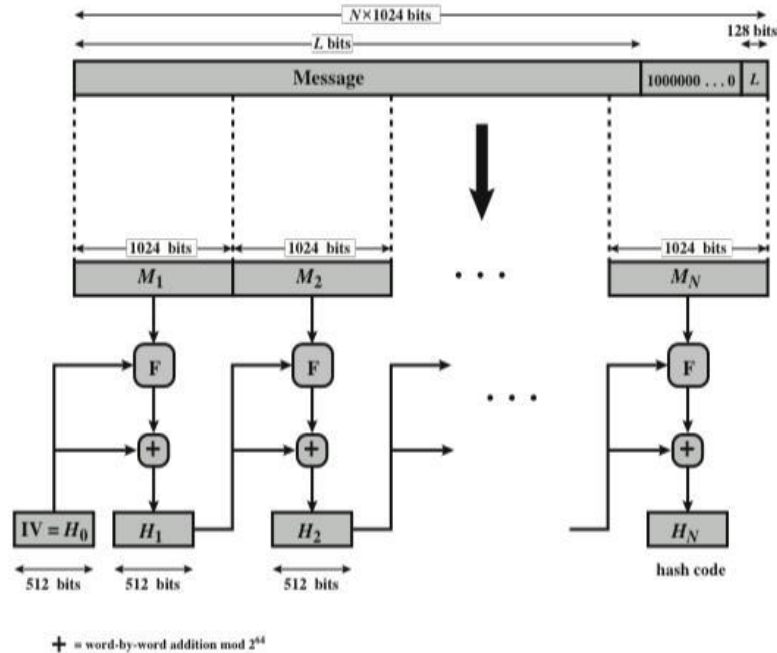


Figure 11.9 Message Digest Generation Using SHA-512

**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

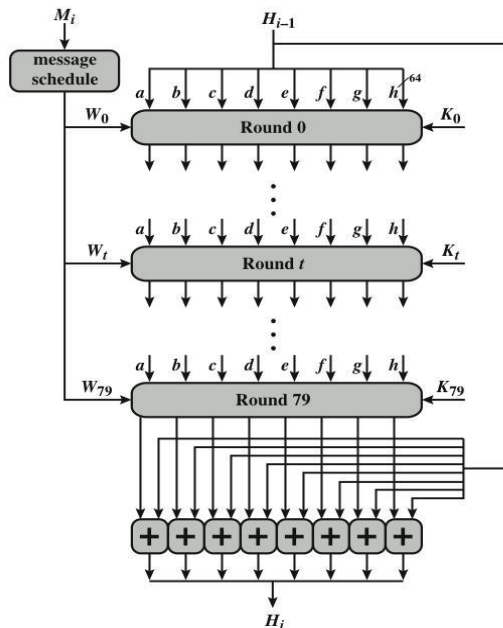


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block



### 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:

$Ch(e,f,g) = (e \text{ AND } f) \text{ XOR } (\text{NOT } e \text{ AND } g)$

$Maj(a,b,c) = (a \text{ AND } b) \text{ XOR } (a \text{ AND } c) \text{ XOR } (b \text{ AND } c)$

$\Sigma(a) = \text{ROTR}(a,28) \text{ XOR } \text{ROTR}(a,34) \text{ XOR } \text{ROTR}(a,39)$

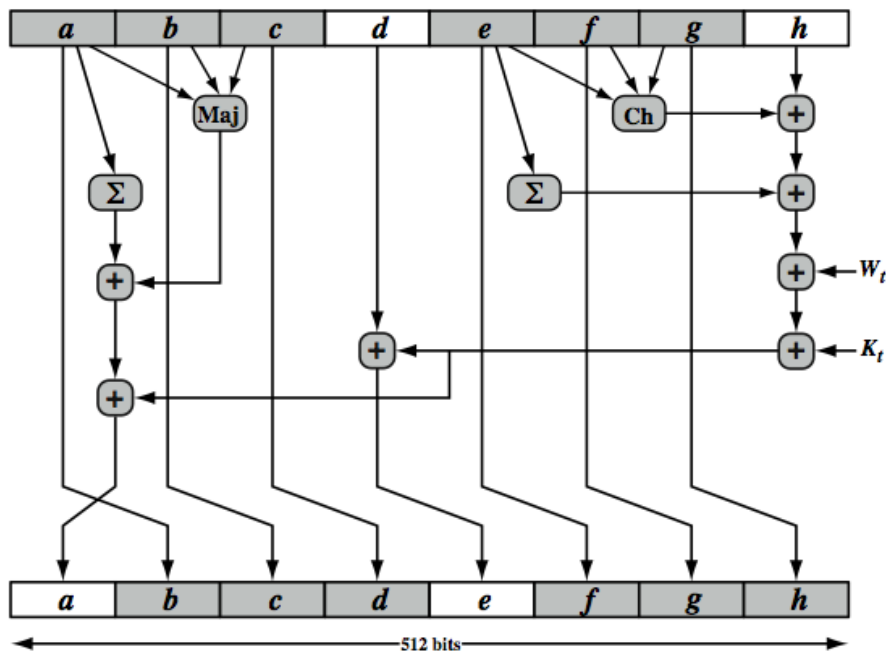
$\Sigma(e) = \text{ROTR}(e,14) \text{ XOR } \text{ROTR}(e,18) \text{ XOR } \text{ROTR}(e,41)$

$+$  = addition modulo  $2^{64}$

$K_t$  = a 64-bit additive constant

$W_t$  = 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  $K_t$ . Word  $a$  is a function of all of the input variables, as well as the round word  $W_t$  and the constant  $K_t$ .



### 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:

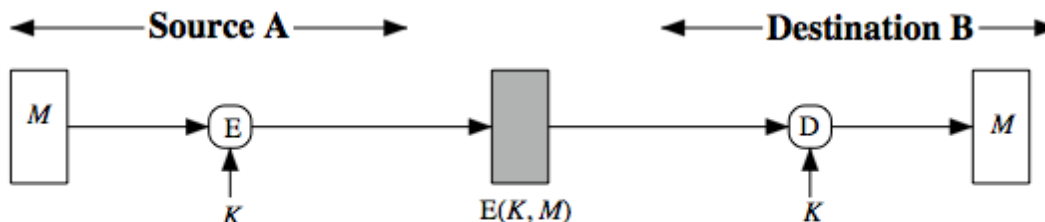
- **hash function:** A function that maps a message of any length into a fixed-length hash value which serves as the authenticator
- **message encryption:** The ciphertext of the entire message serves as its authenticator
- **message authentication code (MAC):** A function of the message and a secret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

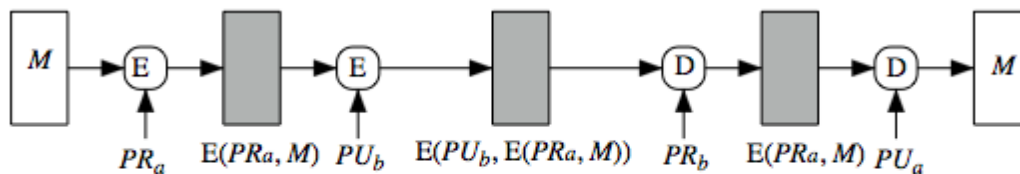


(a) Symmetric encryption: confidentiality and authentication



### 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

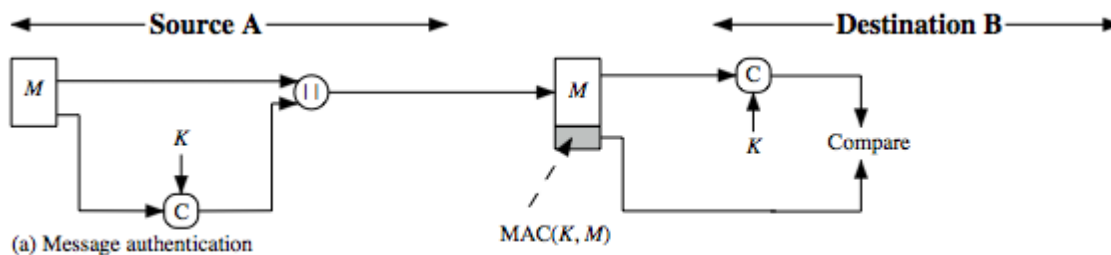


(d) Public-key encryption: confidentiality, authentication, and signature

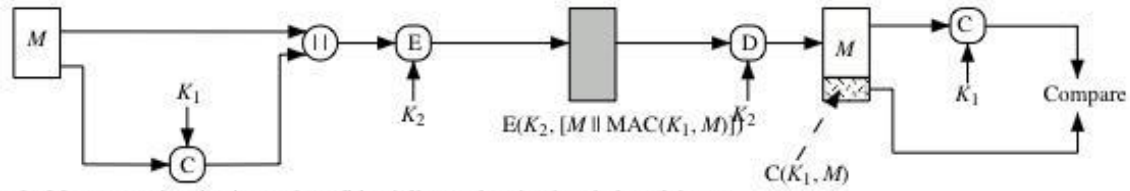
### Message Authentication Code (MAC):

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:  $MAC = C(K, M)$ .

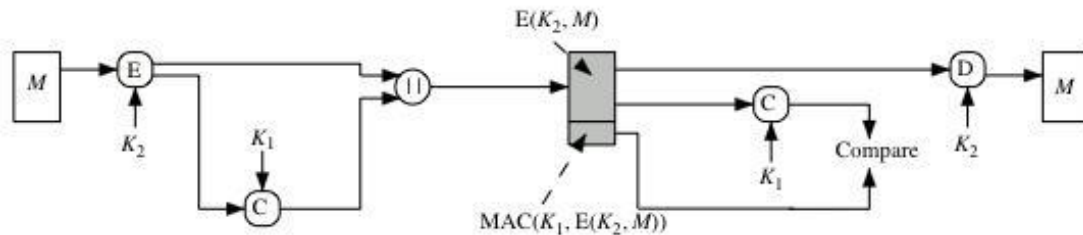
The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC Figure a. If we assume that only the receiver and the sender know the 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.



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.



(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 mechanism 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:

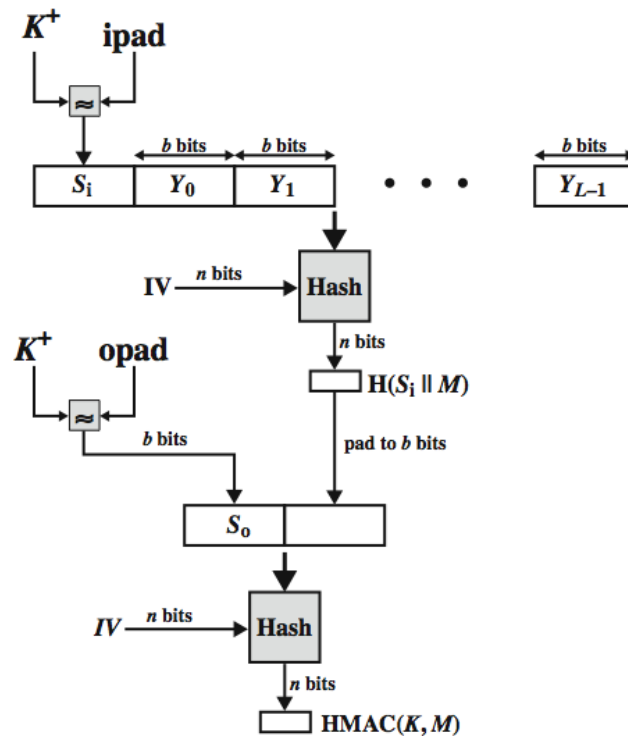
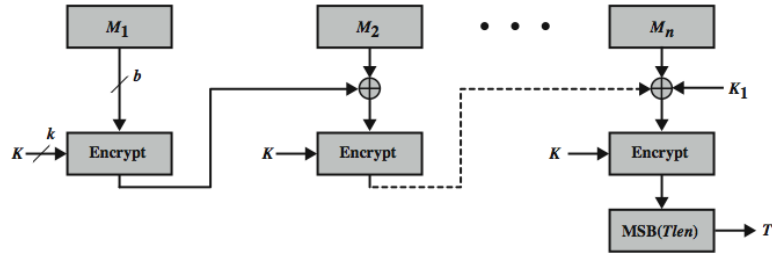


Figure illustrates the overall operation of HMAC

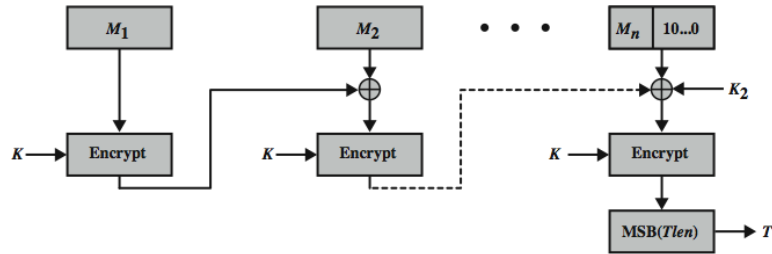
### CMAC:

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



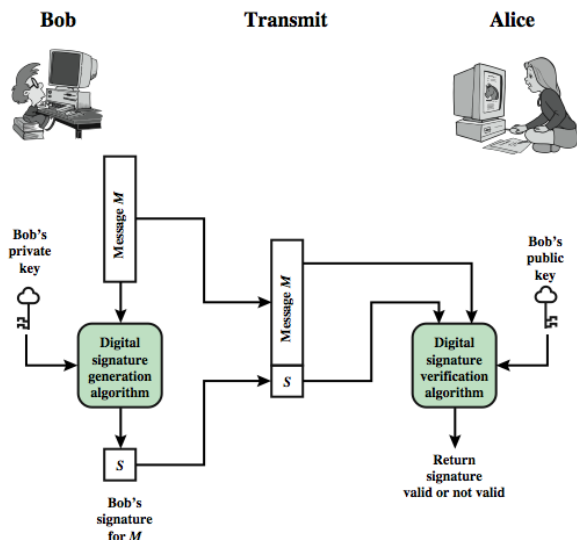
(b) Message length is not integer multiple of block size

**DIGITAL 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:**



**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 break 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.

**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.

**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.

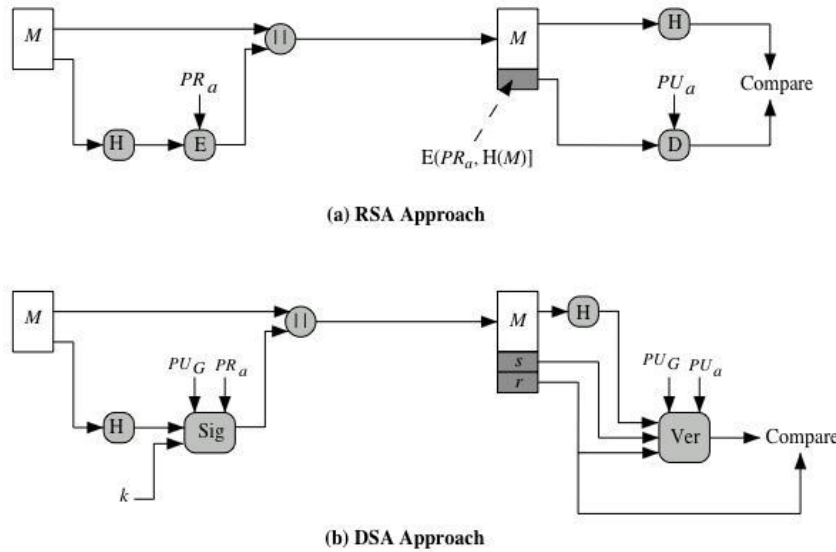


Figure 13.3 Two Approaches to Digital Signatures

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.

**The Digital Signature Algorithm:**

<p style="text-align: center;"><b>Global Public Key Components</b></p> <p><math>p</math> prime number where <math>2^{L-1} &lt; p &lt; 2^L</math> for <math>512 \leq L \leq 1024</math> and <math>L</math> a multiple of 64 i.e., bit length of between 512 and 1024 bits in increments of 64 bits</p> <p><math>q</math> prime divisor of <math>(p - 1)</math>, where <math>2^{159} &lt; q &lt; 2^{160}</math> i.e., bit length of 160 bits</p> <p><math>g = h^{(p-1)/q} \text{ mod } p</math> where <math>h</math> is any integer with <math>1 &lt; h &lt; (p - 1)</math> such that <math>h^{(p-1)/q} \text{ mod } p &gt; 1</math></p> <hr/> <p style="text-align: center;"><b>User's Private Key</b></p> <p><math>x</math> random or pseudorandom integer with <math>0 &lt; x &lt; q</math></p> <hr/> <p style="text-align: center;"><b>User's Public Key</b></p> <p><math>y = g^x \text{ mod } p</math></p> <hr/> <p style="text-align: center;"><b>User's Per-Message Secret Number</b></p> <p><math>k = \text{random or pseudorandom integer with } 0 &lt; k &lt; q</math></p>	<p style="text-align: center;"><b>Signing</b></p> <p><math>r = (g^k \text{ mod } p) \text{ mod } q</math></p> <p><math>s = [k^{-1}(H(M) + xr)] \text{ mod } q</math></p> <p>Signature = <math>(r, s)</math></p> <hr/> <p style="text-align: center;"><b>Verifying</b></p> <p><math>w = (s^{-1}) \text{ mod } q</math></p> <p><math>u_1 = [H(M')w] \text{ mod } q</math></p> <p><math>u_2 = (r')w \text{ mod } q</math></p> <p><math>v = [(g^{u_1} y^{u_2}) \text{ mod } p] \text{ mod } q</math></p> <p>TEST: <math>v = r'</math></p> <p><math>M</math> = message to be signed  <math>H(M)</math> = hash of <math>M</math> using SHA-1  <math>M', r', s'</math> = received versions of <math>M, r, s</math></p>
--	--

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  $h^{(p-1)/q} \bmod p$  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.

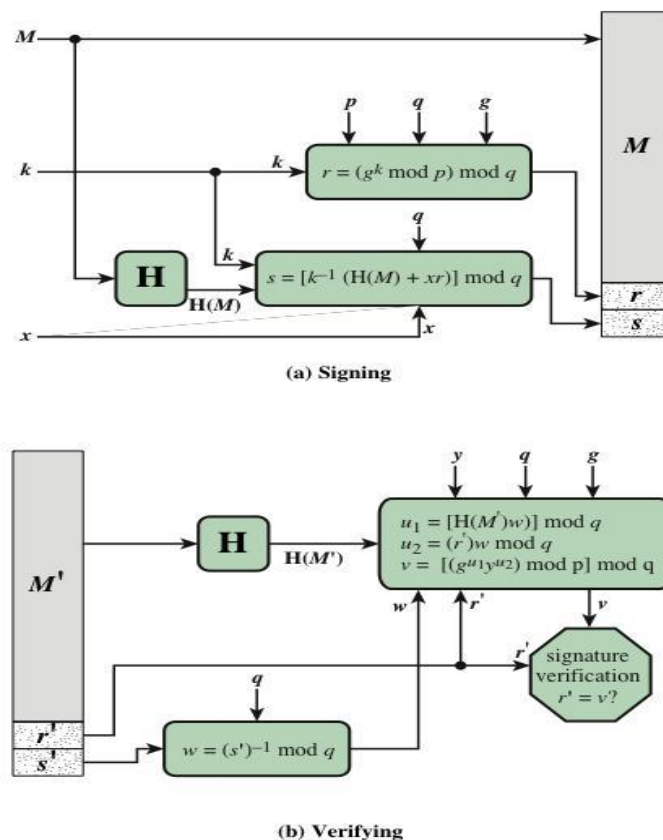


Figure 13.5 DSA Signing and Verifying

#### 4.11.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

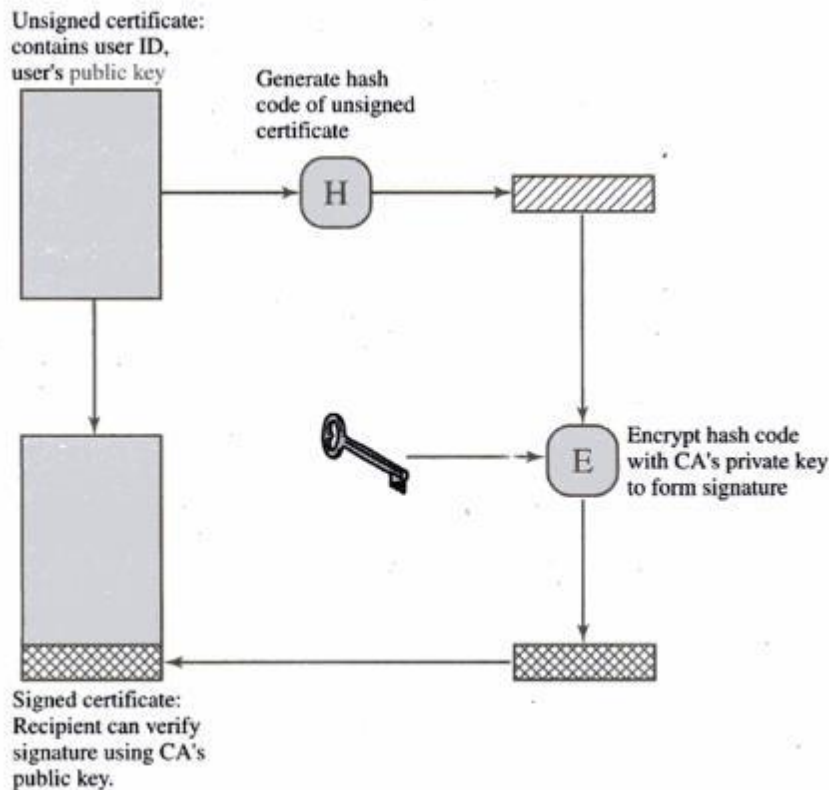


### Public-Key Certificates

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.



---

**UNIT-V****User Authentication:**

In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. User authentication is the basis for most type of access control and for user accountability. RFC 4949 (Internet Security Glossary) defines user authentication.

A typical item of authentication information associated with this user ID is a password, which is kept secret (known only to Alice and to the system). If no one is able to obtain or guess Alice's password, then the combination of Alice's user ID and password enables administrators to set up Alice's access permissions and audit her activity. Because Alice's ID is not secret, system users can send her e-mail, but because her password is secret, no one can pretend to be Alice.

**The process of verifying an identity claimed by or for a system entity is called authentication.**

An authentication process consists of two steps:

- **Identification step:** Presenting an identifier to the security system.
- **Verification step:** Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

In essence, identification is the means by which a user provides a claimed identity to the system; user authentication is the means of establishing the validity of the claim.

**Means of User Authentication:**

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

- **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.

All of these methods, properly implemented and used, can provide secure user authentication. However, each method has problems.

- An adversary may be able to guess or steal a password.
- Similarly, an adversary may be able to forge or steal a token.
- A user may forget a password or lose a token.
- Furthermore, there is a significant administrative overhead for managing password and token information on systems and securing such information on systems.
- With respect to biometric authenticators, there are a variety of problems, including dealing with false positives and false negatives, user acceptance, cost, and convenience.



**Mutual Authentication:**

Protocols which enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

Central to the problem of authenticated key exchange are **two issues**:

- **confidentiality** and
- **timeliness**.

**Confidentiality:** To prevent **masquerade** and to prevent compromise of session keys, essential identification and session-key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose.

**Timeliness**, is important because of the threat of message **replays**. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

**Replay Attacks:**

lists the following examples of replay attacks:

1. The simplest replay attack is one in which the opponent simply copies a message and replays it later.
2. An opponent can replay a timestamped message within the valid time window. If both the original and the replay arrive within then time window, this incident can be logged.
3. As with example (2), an opponent can replay a timestamped message within the valid time window, but in addition, the opponent suppresses the original message. Thus, the repetition cannot be detected.
4. Another attack involves a backward replay without modification. This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

**Approaches to Coping with Replay Attacks:**

1. Attach a **sequence number** to each message used in an authentication exchange
  - A new message is accepted only if its sequence number is in the proper order
  - Difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with
  - Generally, not used for authentication and key exchange because of overhead
2. Timestamps
  - Requires that clocks among the various participants be synchronized
  - Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time
3. Challenge/response
  - Party A, expecting a fresh message from B, first sends B a **nonce** (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value

**One-Way Authentication:**

One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead, the e-mail message is forwarded to the receiver's



---

electronic mailbox, where it is buffered until the receiver is available to read it.

The “envelope” or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP). However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail -handling mechanism.

### **Kerberos:**

Kerberos is an authentication service developed as part of Project Athena at MIT, and is one of the best known and most widely implemented trusted third party key distribution systems.

A workstation cannot be trusted to identify its users correctly to network services

- 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

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption. Two versions of Kerberos are in common use: version 4 & version 5.

### **Kerberos Requirements:**

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, 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 Version 4:**

Makes use of DES to provide the authentication service

There are different approaches to security are:

- 1 SIMPLE AUTHENTICATION DIALOGUE
- 2 MORE SECURE AUTHENTICATION DIALOGUE.

### ***A SIMPLE AUTHENTICATION DIALOGUE:***

- For a secure transaction, server should confirm the client and its request.
- In unprotected network it creates burden on server, therefore an Authentication Server(AS) is used.



- Authentication server (AS)
  - ✓ Knows the passwords of all users and stores these in a centralized database
  - ✓ Shares a unique secret key with each server
- Ticket
  - ✓ Created once the AS accepts the user as authentic; contains the user's ID and network address and the server's ID
  - ✓ Encrypted using the secret key shared by the AS and the server

Consider the following hypothetical dialogue

$$(1) C \rightarrow AS: ID_C || P_C || ID_V$$

$$(2) AS \rightarrow C: Ticket$$

$$(3) C \rightarrow V: ID_C || Ticket$$

$$Ticket = E(K_v, [ID_C || AD_C || ID_V])$$

where

C = client

AS = authentication server

V = server

$ID_C$  = identifier of user on C

$ID_V$  = identifier of V

$P_C$  = password of user on C

$AD_C$  = network address of C

$K_v$  = secret encryption key shared by AS and V

**Problem:** An opponent could capture the ticket transmitted in message (2), then use the name  $ID_C$  and transmit a message of form (3) another workstation. The server would receive a valid ticket that matches the user ID and grant access to the user on that other workstation. To prevent this attack, the AS includes in the ticket the network address from which the original request came.

**A MORE SECURE AUTHENTICATION DIALOGUE:**

- ✓ The main problem in A SIMPLE AUTHENTICATION DIALOGUE, the user must enter password for every individual service.
- ✓ Kerberos overcome this by using a new server, known as **Ticket granting server (TGS)**.
- ✓ Now in Kerberos we have two servers; **AS and TGS**.

**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:**

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

$$Ticket_{tgs} = E(K_{tgs}, [ID_C || AD_C || ID_{tgs} || TS_1 || Lifetime_1])$$

$$Ticket_v = E(K_v, [ID_C || AD_C || ID_V || TS_2 || Lifetime_2])$$



The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.

Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password, which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.
3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
4. The TGS decrypts the incoming ticket using a key shared only by the AS and the TGS and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.
5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service -granting ticket. The server authenticates by using the contents of the ticket.

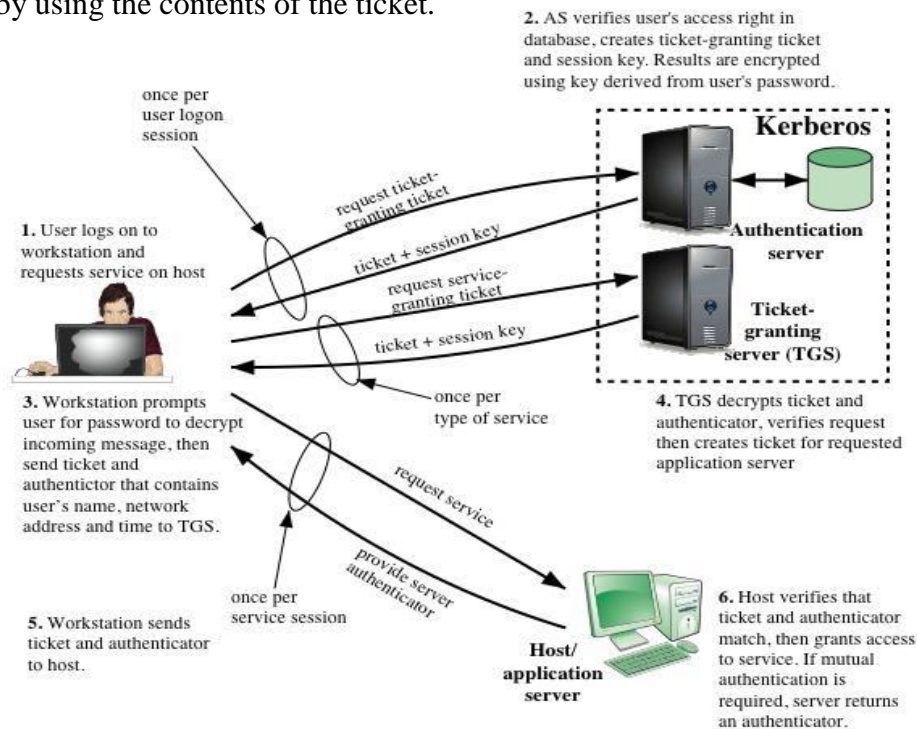


Figure 15.1 Overview of Kerberos



### KERBEROS REALMS:

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers 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

A **Kerberos realm is a set of managed nodes that share the same Kerberos database.** The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room. A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

### Kerberos Version 5:

Kerberos version 5 is specified in RFC 4120 and provides a number of improvements over version 4. Kerberos version 4 was developed for use within the Project Athena environment and, accordingly, did not fully address the need to be of general purpose.

### Kerberos version 5 Authentication Dialogue:

The Kerberos version 5 message exchange involves three sessions, these are

- a) Authentication service exchange
- b) Ticket-granting exchange
- c) Client/server authentication exchange.

Each session has two steps.

(1) C → AS Options || IDc || Realmc || IDtgs || Times || Nonce1  
 (2) AS → C Realmc || IDC || Tickettgs || E(Kc, [Kc,tgs || Times || Nonce1 || Realmtgs || IDtgs])  
 Tickettgs = E(Ktgs, [Flags || Kc,tgs || Realmc || IDC || ADC || Times])

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C → TGS Options || IDv || Times || Nonce2 || Tickettgs || Authenticatorc  
 (4) TGS → C Realmc || IDC || Tickettv || E(Kc,tgs, [Kc,v || Times || Nonce2 || Realmv || IDv])  
 Tickettgs = E(Ktgs, [Flags || Kc,tgs || Realmc || IDC || ADC || Times])  
 Tickettv = E(Kv, [Flags || Kc,v || Realmc || IDC || ADC || Times])  
 Authenticatorc = E(Kc,tgs, [IDC || Realmc || TS1])

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C → V Options || Tickettv || Authenticatorc  
 (6) V → C E(Kc,v, [TS2 || Subkey || Seq#])  
 Tickettv = E(Kv, [Flags || Kc,v || Realmc || IDC || ADC || Times])  
 Authenticatorc = E(Kc,v, [IDC || Realmc || TS2 || Subkey || Seq#])

(c) Client/Server Authentication Exchange to obtain service



**Environmental differences between Kerberos version 4 and version 5:**

S.No	Parameters	Version 4	Version 5
1	Encryption system dependence	Version 4 requires the use of DES.	In version 5, ciphertext is tagged with an encryption-type identifier so that any encryption technique may be used.
2	Internet protocol dependence	Version 4 requires the use of Internet Protocol (IP) addresses.	Version 5 network addresses are tagged with type and length, allowing any network address type to be used.
3	Message byte ordering	In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address.	In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.
4	Ticket lifetime	Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is $28 \times 5 = 1280$ minutes. This may be inadequate for some applications.	In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.
5	Authentication forwarding	Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client.	Version 5 provides this capability authentication forwarding.
6	Inter-realm authentication	In version 4, interoperability among N realms requires on the order of $N^2$ Kerberos-to-Kerberos relationships, as described earlier.	Version 5 supports a method that requires fewer relationships.



**Technical differences between Kerberos version 4 and version 5:**

S.No	Parameters	Version 4	Version 5
1	Double encryption	In version 4, that tickets provided to clients are encrypted twice—once with the secret key of the target server and then again with a secret key known to the client.	In version 5, the ticket which are issued to the clients are encrypted with only one key.
2	PCBC encryption	Encryption in version 4 makes use of a nonstandard mode of DES known as propagating cipher block chaining (PCBC).	Version 5 provides explicit integrity mechanisms, allowing the standard CBC mode to be used for encryption
3	Session keys	The session keys are included in each ticket that can be later used by client and server.	Each time connection is established a different sub session key is used by both client and server.
4	Password attacks	It does not provide any mechanism to prevent attacks on passwords	Version 5 provides a mechanism known as pre authentication but it does not prevent password attacks

**Transport Level Security:**

**Web security considerations:**

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets

The following characteristics of Web usage suggest the need for tailored security tools:

- Web servers are relatively easy to configure and manage
- Web content is increasingly easy to develop
- The underlying software is extraordinarily complex
  - ✓ May hide many potential security flaws



- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex
- Casual and untrained (in security matters) users are common clients for Web-based services
  - ✓ Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures

### Web security Threats:

	Threats	Consequences	Countermeasures
<b>Integrity</b>	<ul style="list-style-type: none"> <li>•Modification of user data</li> <li>•Trojan horse browser</li> <li>•Modification of memory</li> <li>•Modification of message traffic in transit</li> </ul>	<ul style="list-style-type: none"> <li>•Loss of information</li> <li>•Compromise of machine</li> <li>•Vulnerability to all other threats</li> </ul>	Cryptographic checksums
<b>Confidentiality</b>	<ul style="list-style-type: none"> <li>•Eavesdropping on the net</li> <li>•Theft of info from server</li> <li>•Theft of data from client</li> <li>•Info about network configuration</li> <li>•Info about which client talks to server</li> </ul>	<ul style="list-style-type: none"> <li>•Loss of information</li> <li>•Loss of privacy</li> </ul>	Encryption, Web proxies
<b>Denial of Service</b>	<ul style="list-style-type: none"> <li>•Killing of user threads</li> <li>•Flooding machine with bogus requests</li> <li>•Filling up disk or memory</li> <li>•Isolating machine by DNS attacks</li> </ul>	<ul style="list-style-type: none"> <li>•Disruptive</li> <li>•Annoying</li> <li>•Prevent user from getting work done</li> </ul>	Difficult to prevent
<b>Authentication</b>	<ul style="list-style-type: none"> <li>•Impersonation of legitimate users</li> <li>•Data forgery</li> </ul>	<ul style="list-style-type: none"> <li>•Misrepresentation of user</li> <li>•Belief that false information is valid</li> </ul>	Cryptographic techniques

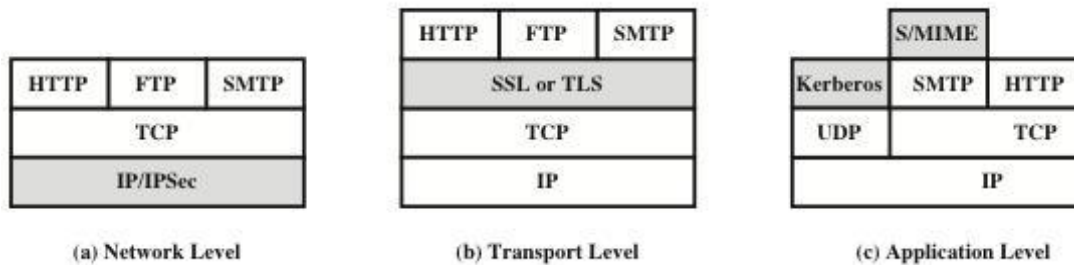
**Table. A Comparison of Threats on the Web**

### Web Traffic Security Approaches:

A number of approaches to providing Web security are possible.

1. One way to provide Web security is to use **IP security (IPsec)** (Figure(a)). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution.
2. Another relatively general-purpose solution is to implement security just above TCP (Figure (b)). The foremost example of this approach is the **Secure Sockets Layer (SSL)** and the follow-on Internet standard known as **Transport Layer Security (TLS)**. At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example, Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.
3. Application-specific security services are embedded within the particular application. Figure (c) shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.





*Figure: relative location of security facilities in the TCP/IP Protocol stack*

### **SSL (Secure Socket Layer):**

SSL probably most widely used Web security mechanism, and it is implemented at the Transport layer.

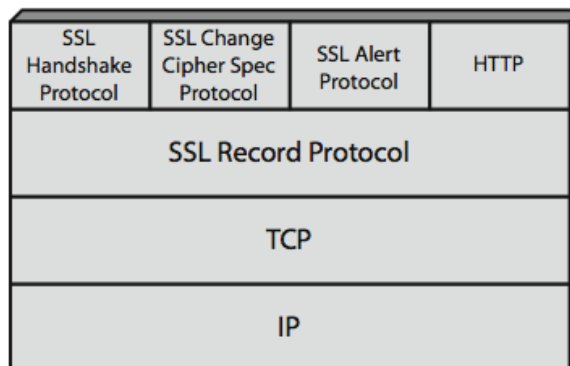
- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- Netscape originated SSL. Version 3 of the protocol was designed with public review and input from industry and was published as an Internet draft document. Subsequently, became Internet standard known as TLS (Transport Layer Security)

### **SSL Architecture:**

- SSL is designed to make use of TCP to provide a reliable end-to-end secure service.
- SSL is not a single protocol but rather two layers of protocols.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

1. **Connection:** A connection is a transport that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. Every connection is associated with one session.
2. **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections.



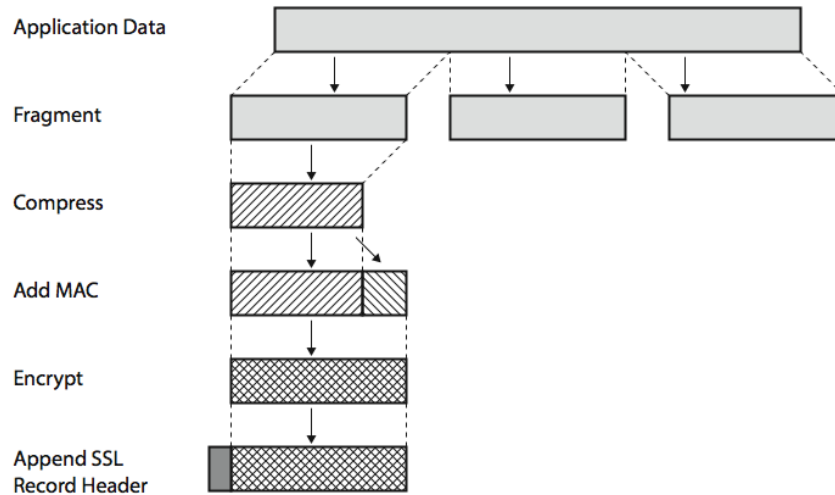
*Figure: SSL Protocol stack*



### SSL Record Protocol:

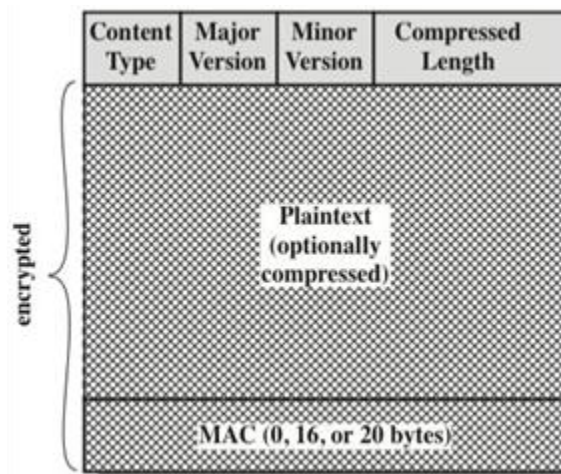
SSL Record Protocol defines two services for SSL connections:

1. **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads. The message is compressed before being concatenated with the MAC and encrypted, with a range of ciphers being supported as shown.
2. **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).



**Figure: SSL Record Protocol Operation**

Figure shows the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.



**Figure: SSL Record Format**



The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:

- **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
- **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
- **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
- **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is  $2^{14} + 2048$ .

### ChangeCipherSpec Protocol:

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

1 byte

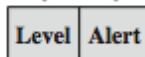


(a) Change Cipher Spec Protocol

### SSL Alert Protocol:

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state. Each message in this protocol consists of two bytes, the first takes the value warning (1) or fatal (2) to convey the severity of the message. The second byte contains a code that indicates the specific alert.

1 byte 1 byte



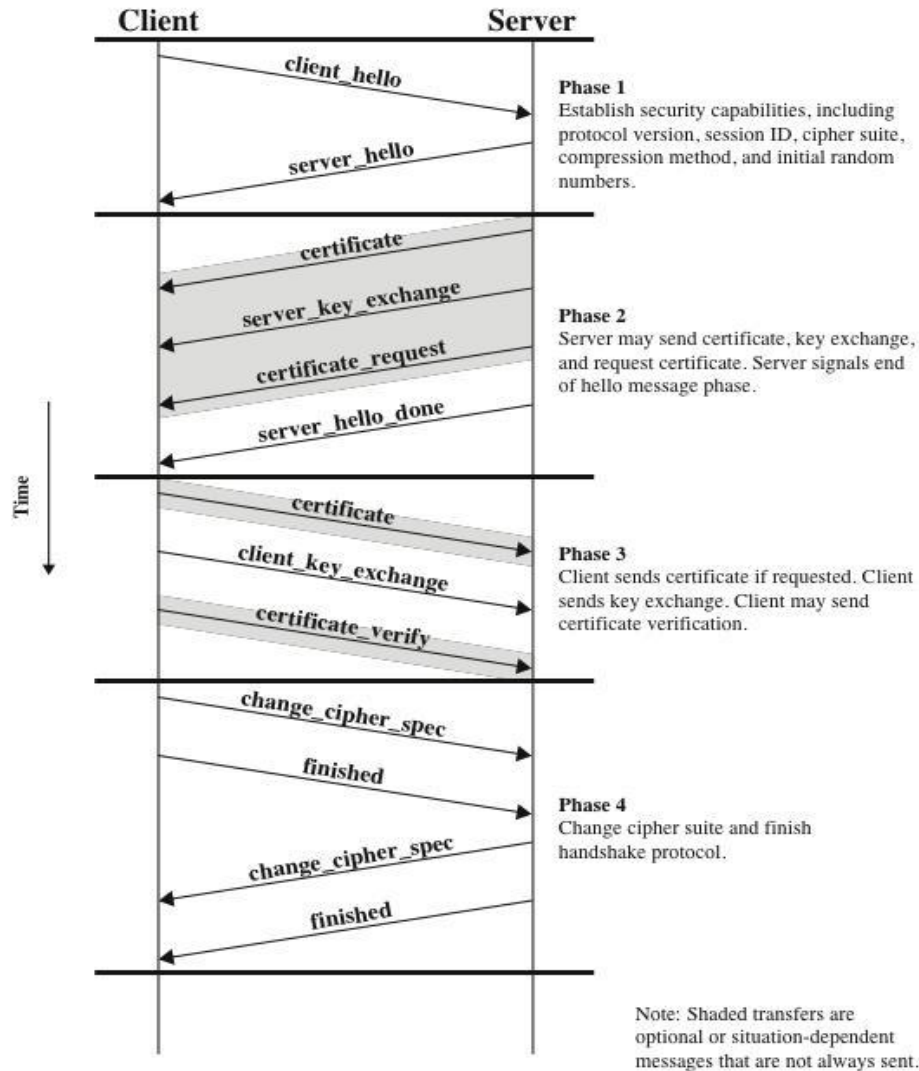
(b) Alert Protocol

### SSL Handshake Protocol:

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client to authenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted. The Handshake Protocol consists of a series of messages exchanged by client and server.



(c) Handshake Protocol



**Figure 17.6 Handshake Protocol Action**

### Transport Layer Security (TLS) Protocol

In order to provide an open Internet standard of SSL, Internet Engineering Task Force(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.



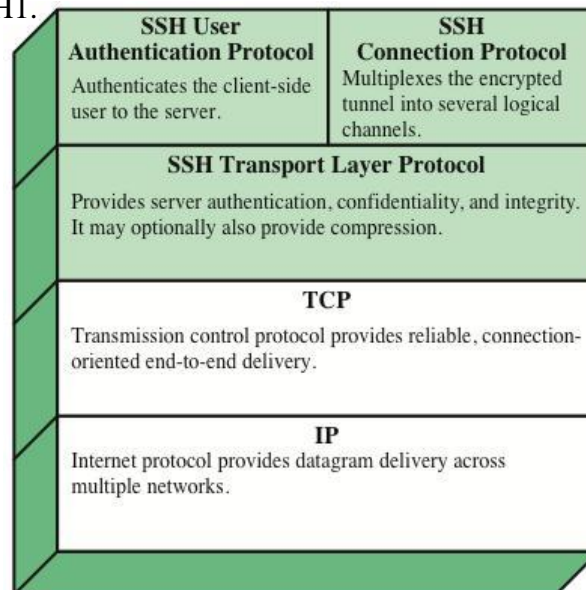
### Comparison of TLS and SSL Protocols:

1. **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.
2. **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.
3. **Session Key Generation** – There are two differences between TLS and SSL protocol for generation of key material.
  - I. 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.
  - II. The algorithm for computing session keys and initiation values (IV) is different in TLS than SSL protocol.
4. **Alert Protocol Message** –
  - I. 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.
  - II. Many additional Alert messages are included in TLS protocol for other error conditions such as *record\_overflow*, *decode\_errortc*.

### 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: SSH Protocol stack*



### **Transport Layer Protocol:**

In this part of SSH protocol provides data confidentiality, server (host) authentication, and data integrity. It may optionally provide data compression as well.

- **Server Authentication** – Host keys are asymmetric like public/private keys. A server uses a public key to prove its identity to a client. The client verifies that contacted server is a “known” host from the database it maintains. Once the server is authenticated, session keys are generated.
- **Session Key Establishment** – After authentication, the server and the client agree upon cipher to be used. Session keys are generated by both the client and the server. Session keys are generated before user authentication so that usernames and passwords can be sent encrypted. These keys are generally replaced at regular intervals (say, every hour) during the session and are destroyed immediately after use.
- **Data Integrity** – SSH uses Message Authentication Code (MAC) algorithms to for data integrity check. It is an improvement over 32 bit CRC used by SSH1.

### **User Authentication Protocol:**

In this part of SSH authenticates the user to the server. The server verifies that access is given to intended users only. Many authentication methods are currently used such as, typed passwords, Kerberos, public-key authentication, etc.

### **Connection Protocol:**

This provides multiple logical channels over a single underlying SSH connection

### **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.

### **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

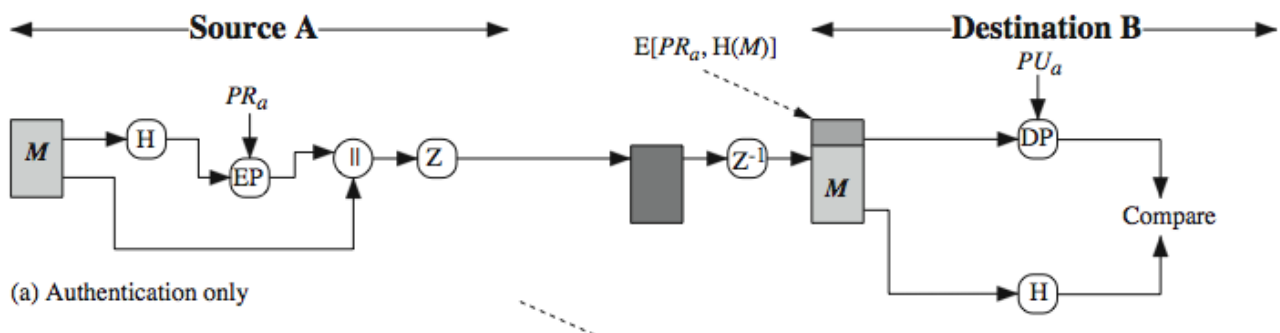
- ✓ Selected the best available cryptographic algorithms as building blocks
- ✓ Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands
- ✓ Made the package and its documentation, including 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

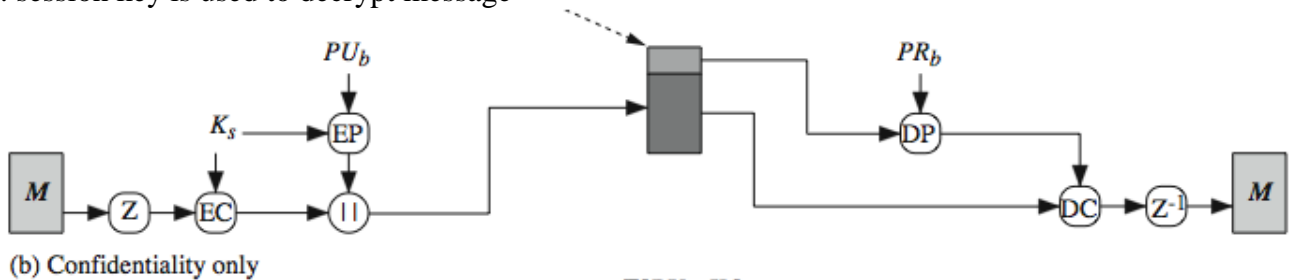


### PGP Operation – Confidentiality:

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

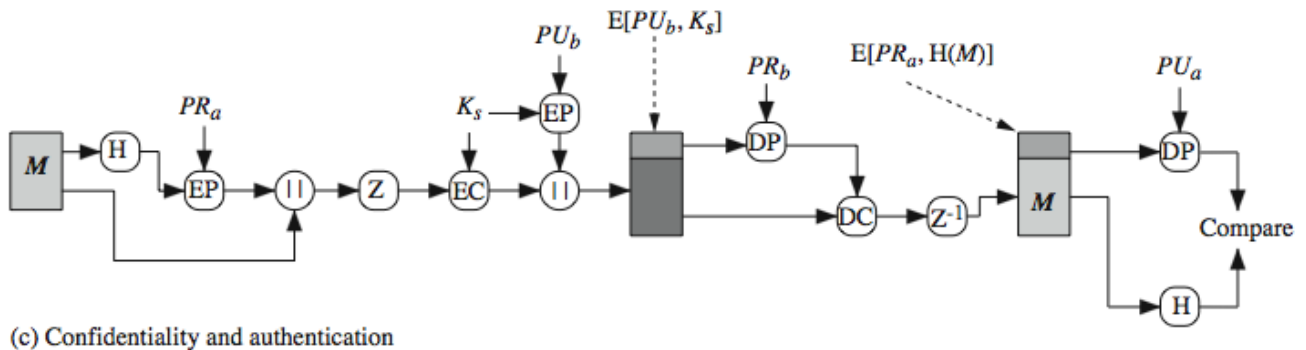


3. session key is encrypted using RSA with recipient's public key, then attached to message
4. receiver uses RSA with its private key to decrypt and recover session key
5. session key is used to decrypt message



### PGP Operation – Confidentiality & Authentication:

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



### 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 fields that 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 multimedia electronic mail.
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 represent the 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 signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with 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 without S/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 between a 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



---

## UNIT-VI

### 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 while in transit.

### IPSEC SECURITY FEATURES:

IPSec is the most secure method commercially available for connecting network sites. IPSec was designed 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:

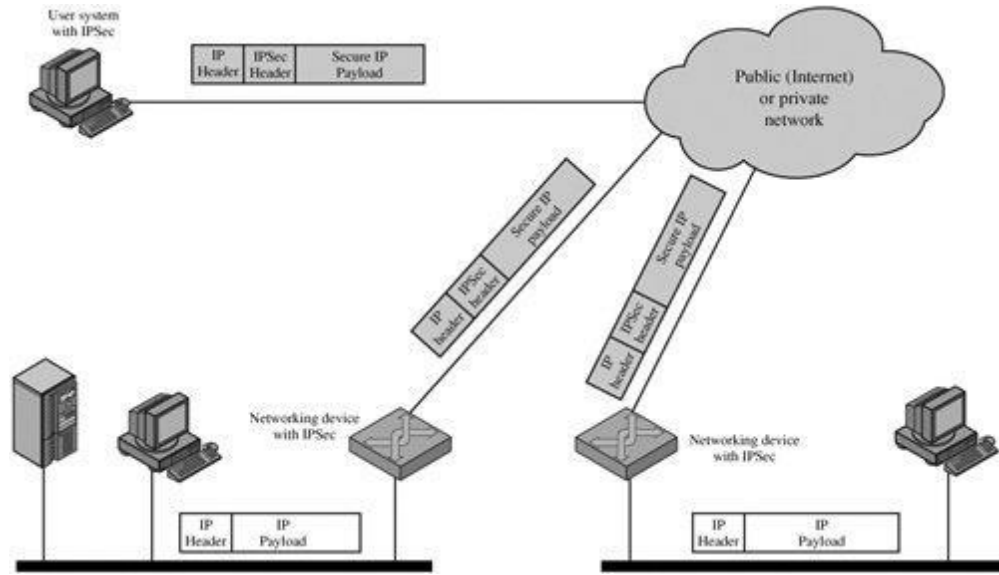
IPSec contains the following elements:

- **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 a key 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:**

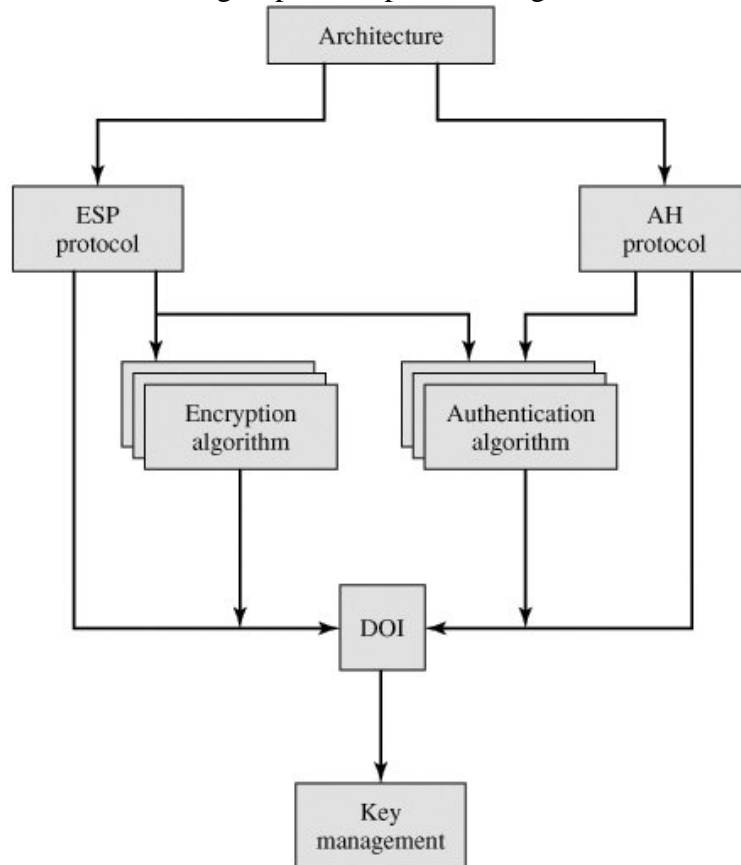
The IPsec specification consists of numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408:

- 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 IPsec technology.
- **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 the use 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 approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.



### IPSec Services:

IPSec provides security services at the IP layer by selecting required security protocols, algorithms and cryptographic keys 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 not both.

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 associations is 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 used to generate the Sequence Number field in AH or ESP headers.



- **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA.
- **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.

### 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 to be 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 forwarded and 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 SPD to 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.
- **User ID:** 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 description of AH and ESP.

### Transport Mode:

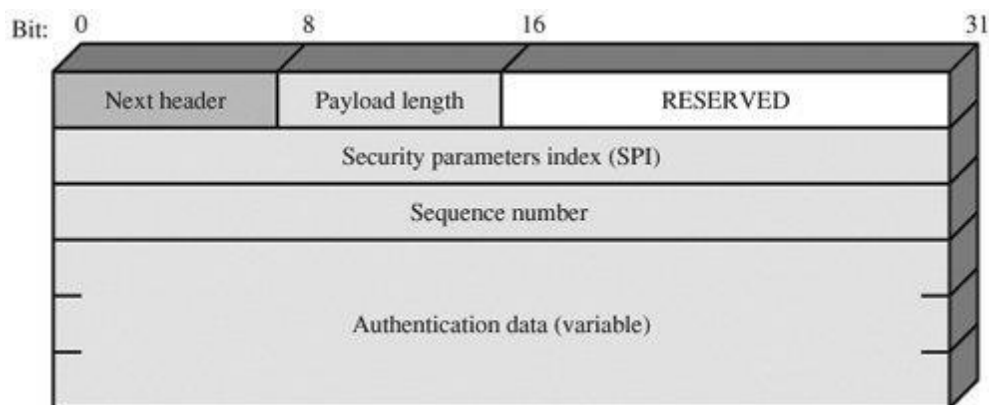
Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet. Transport mode is used for end-to-end communication between two hosts. ESP in transport mode encrypts and optionally authenticates the IP payload but not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

### **AUTHENTICATION HEADER(AH):**

The Authentication Header provides support for data integrity and authentication of IP packets. Data integrity 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.



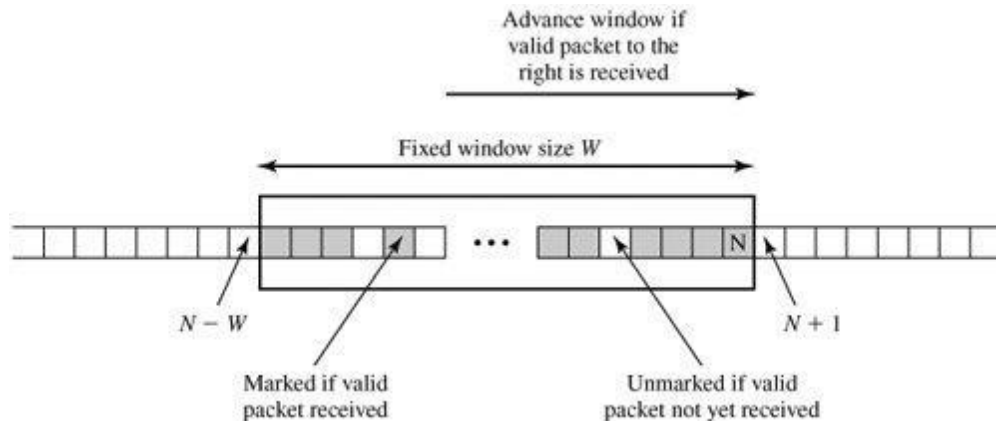


5. **Sequence Number (32 bits):** A monotonically increasing counter value.

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 intended destination. The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence. The **Sequence Number** field is designed to stop such attacks.



*Figure. Antireplay Mechanism*

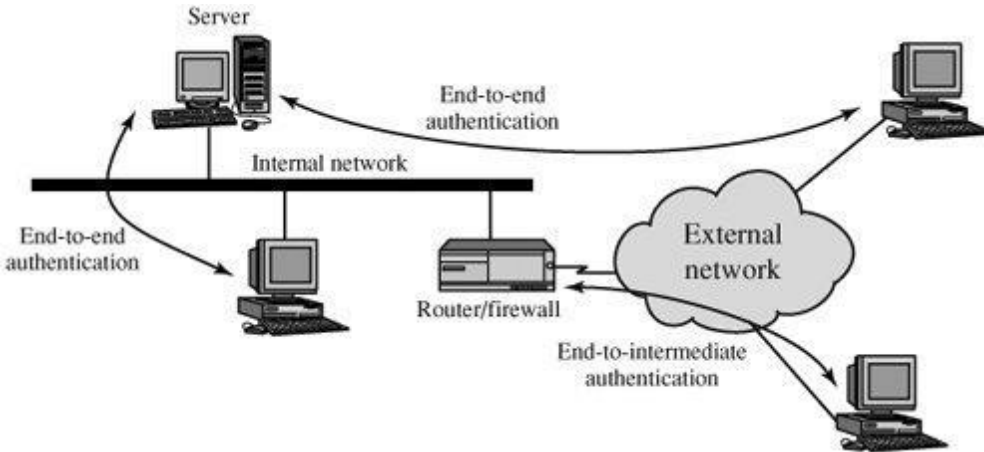
When a new SA is established, the sender initializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. If anti-replay is enabled (the default), the sender must not allow the sequence number to cycle past 232-1 back to zero. Otherwise, there would be multiple valid packets with the same sequence number. If the limit of 232-1 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.

### **Integrity Check Value:**

The Authentication Data field holds a value referred to as the Integrity Check Value. The ICV is a message 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 authentication service can be used. In one case, **authentication is 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 share a 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 access to 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*

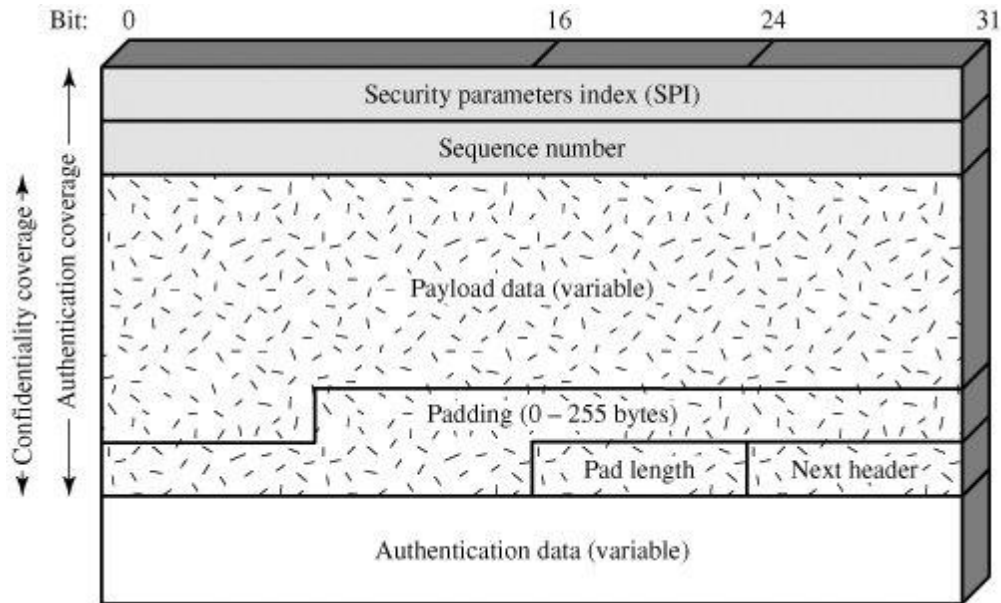
### **Encapsulating Security Payload(ESP):**

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

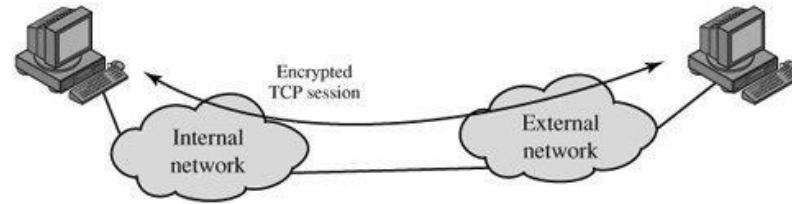
### **Padding:**

The Padding field serves several purposes:

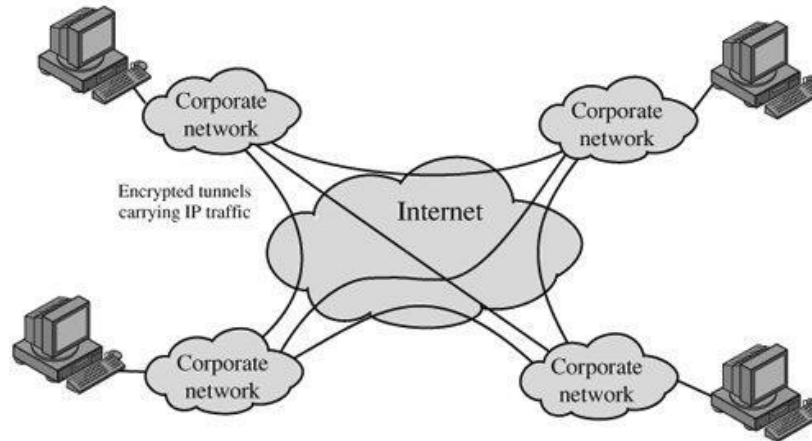
1. If an encryption algorithm requires the plaintext to be a multiple of some number of bytes. The Padding field is used to expand the plaintext to the required length.
2. The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext 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 allows the hosts to avoid implementing the security capability. The former technique is support by a transport mode SA, while the latter technique uses a tunnel model SA.



(a) Transport-level security



(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 provided by both AH and ESP. Further, a particular traffic flow may require 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 sequence of SAs through which traffic must 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 IP packet, without invoking tunneling.
- **Iterated tunneling:** Refers to the application of multiple layers of security protocols effected through 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 with the 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 and consists 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 based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate 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

### **Features of Oakley:**

The Oakley algorithm is characterized by five important features:

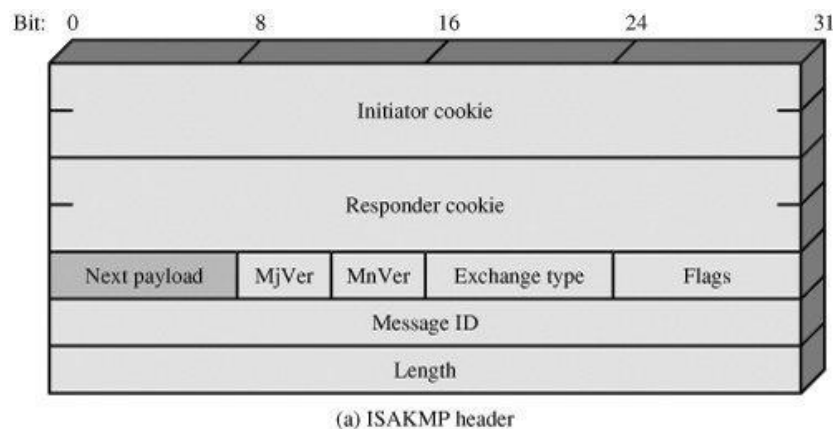
- It employs a mechanism known as cookies to thwart clogging attacks.
- It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of the Diffie-Hellman key exchange.
- It uses nonces to ensure against replay attacks.
- It enables the exchange of Diffie-Hellman public key 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 protocol support, 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 use of UDP for the transport protocol.





It consists of the following fields:

1. **Initiator Cookie (64 bits):** Cookie of entity that initiated SA establishment, SA notification, or SA 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.

### Intrusion Detection/Prevention System:

#### Intrusion

A set of actions aimed to compromise the security goals, namely Integrity, confidentiality, or availability, of a computing and networking resource. It is act of gaining unauthorized access to a system so as to cause loss.

#### Intrusion detection

The process of identifying and responding to intrusion activities

#### Intrusion prevention

Extension of ID with exercises of access control to protect computers from exploitation

#### Terminology:

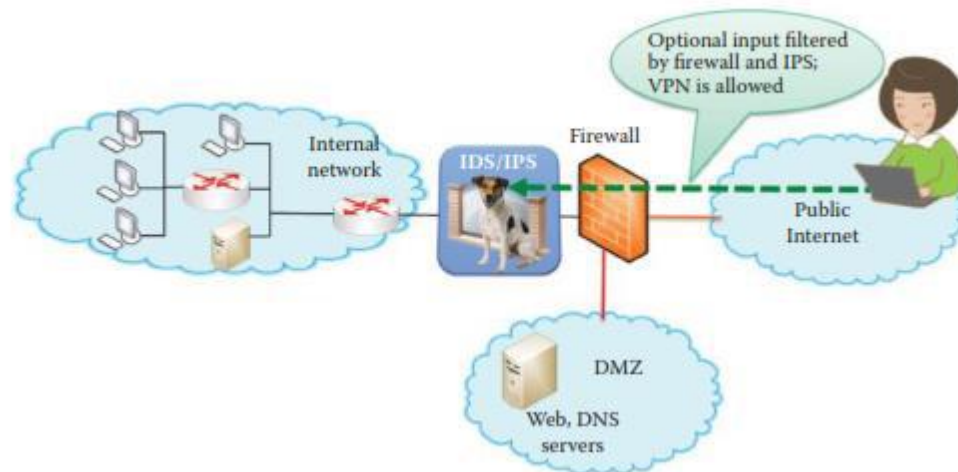
- **True Positives:** These are alerts that something is not right when it is actually not right.
- **True negatives:** these are alerts that something is right when it is actually right.
- **False positives:** these are alerts indicating that something is not right with a packet when actually it is right.
- **False Negatives:** these are alerts that something is right when actually it is wrong.



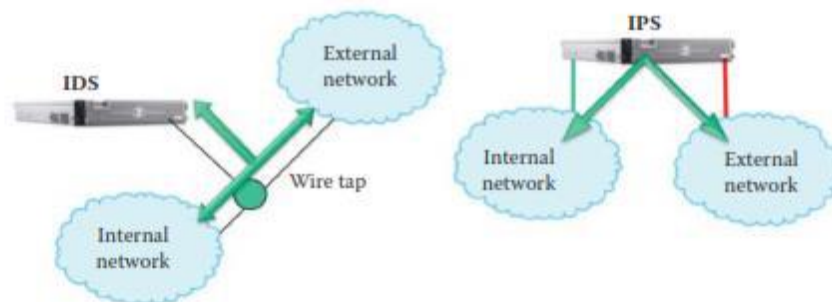
Figure 11-1. Definitions of IDS/IPS Alerts

**OVERVIEW:**

An intrusion detection/prevention system (IDS/IPS) is another element in which it employed to provide deep packet inspection at the entrance of important network. Intrusion Detection System/Intrusion Prevention System is positioned behind the firewall, as shown in Figure.



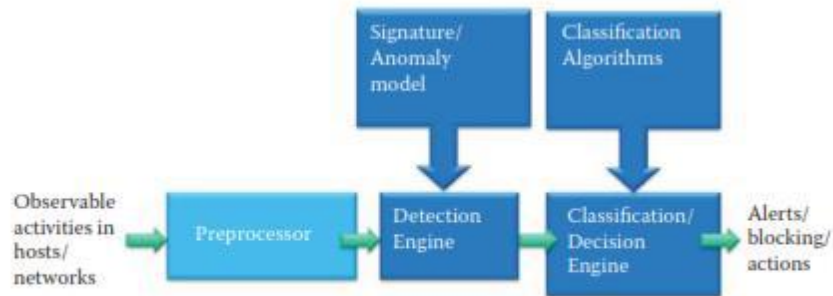
The IDS/IPS provides deep packet inspection for the payload, IDS is based on out-of-band detection of intrusions and their reporting, and IPS is in-band filtering to block intrusions. IDS is performed through a wiretap, and is clearly an out-of-band operation. In contrast, IPS is performed inline. And by preventing intrusions, IPSs eliminate the need for keeping and reading extensive intrusion-incident logs, which contributes to IDSs' considerable CPU, memory, and I/O overhead.



**FIGURE 19.2** An illustration of out-of-band IDS vs. in-band IPS.

**IDS/IPS BUILDING BLOCKS:**

A block diagram that outlines the functions of an IDS/IPS system is shown in Figure. As indicated, the observable activities are preprocessed and forwarded to the detection engine that uses a Signature/Anomaly model. This information is then forwarded to the classification decision engine that uses classification algorithms to provide the alerts or blocking actions.



**FIGURE 19.3** An IDS/IPS system processes activities and generates alerts and blockings.

### **HOST-BASED OR NETWORK-BASED IDS/IPS:**

IDS/IPS can be either **host-based or network-based**, in which case it is labeled as HIDS/HIPS or NIDS/NIPS, respectively. In the HIDS/HIPS case, the **monitoring and blocking activity is performed on a single host**. HIDS/HIPS has the advantage that it provides better visibility into the behavior of individual applications running on that host. HIDS/HIPS monitoring also includes attacks by genuine users/insiders. These include illegitimate use of root privileges; unauthorized access to resources and data. In the NIDS/NIPS, it is often located behind a router or firewall that provides the guarded entrance to a critical asset. At this location traffic is monitored and packet headers and payloads are examined using the knowledge base in NIDS/NIPS. The advantage of this location is that a single NIDS/NIPS can protect many hosts as well as detect global patterns.

There are **various types of IPS products**.

- **Host-based application** firewalls perform the IPS function independently of the operating system and block the entry of application-level and web based intrusions, much like network firewalls bar entry to unwanted traffic.
- A **network-based IPS** blocks network-level intrusions, such as denial-of-service attacks, and may use anomaly detection to recognize threats based on their behavior.
- Combining network- and host-based IPSs provides the best protection against all types of intrusions.

### **THE APPROACHES USED FOR IDS/IPS:**

The approaches to **intrusion detection** can generally be classified as either **anomaly/behavior based** or **signature-based**.

- Anomaly-based detectors generate the normal behavior/pattern of the protected system, and deliver an anomaly alarm if the observed behavior at an instant does not conform to expected behavior.
- Anomaly-based IDS/IPS are more likely to generating false positives due to the dynamic nature of networks, applications and exploits.
- According to the type of processing, anomaly detection techniques can be classified into three main categories: **statistical-based, knowledge-based, and machine learning-based**.

### **STATISTICAL-BASED IDS/IPS:**

In the statistical-based IDS/IPS, the behavior of the system is represented from the **captured network traffic activity** and a profile representing its stochastic behavior is created. This profile is based on





metrics such as the **traffic rate, the number of packets for each protocol, the rate of connections, the number of different IP addresses**, etc. This method employs the collected profile that relates to the behavior of genuine users and is then used in statistical tests to determine if the behavior under detection is genuine or not. During the anomaly detection process, one corresponding to the currently captured profile is compared with the previously trained statistical profile. As the network events occur, the current profile is determined and an anomaly score estimated by comparison of the two behaviors. The score normally indicates the degree of deviation for a specific event.

### Advantages

- First, they do not require prior knowledge about the normal activity of the target system; instead, they have the ability to learn the expected behavior of the system from observations.
- Second, statistical methods can provide accurate notification of malicious activities occurring over long periods of time.

### Drawbacks

- setting the values of the thresholds, parameters/metrics that is a difficult task, especially because the balance between false positives and false negatives is affected.
- Not all behaviors can be modeled by using stochastic methods.

### KNOWLEDGE-/EXPERT-BASED IDS/IPS:

Knowledge-based IDS/IPS captures the **normal behavior from available information, including expert knowledge, protocol specifications, network traffic instances**, etc. The normal behavior is represented as a **set of rules**. Attributes and classes are identified from the training data or specifications. Then a set of classification rules, parameters or procedures are generated. The rules are used for detecting anomaly behaviors. Specification-based anomaly methods require that the model is manually constructed by human experts in terms of a set of rules (the specifications) that describing the system behavior. Specification-based techniques have been shown to produce a low rate of false alarms, but are not as effective as other anomaly detection methods in detecting novel attacks, especially when it comes to network probing and denial-of-service attacks.

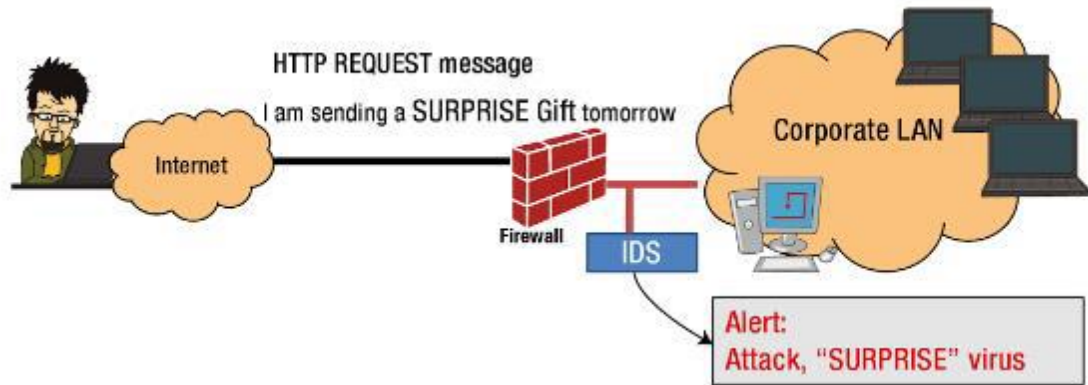
### SIGNATURE BASED IDS:

This mechanism proceeds against known threats. A signature is a known pattern of a threat, such as:

- An e-mail with an attachment containing a malware with an interesting subject.
- A "remote login" by an admin user, which is a clear violation of an organization's policy.

Signature-based detection is the simplest form of detection because it just the traffic with the signature database. If a match found then the alert is generated, if a match is not found then the traffic flows without any problem. In signature-based detection, detection is based on comparing the traffic with the known signatures for possible attacks. They can only detect known threats and hence, are not efficient in detecting unknown threats. To detect an attack, the signature matching has to be precise, otherwise, even if the attack has a small variation from the known threat signature, then the system will not detect. Hence, it is very easy for the attackers to compromise and breach into the trusted network.

Signature database needs to be updated constantly, almost on a daily basis from the anti-virus labs. If the signature is not up to date, chances are that the IDS systems will fail to detect some of the intrusion attacks. The other disadvantage is that they have very little information about previous requests when processing the current ones.

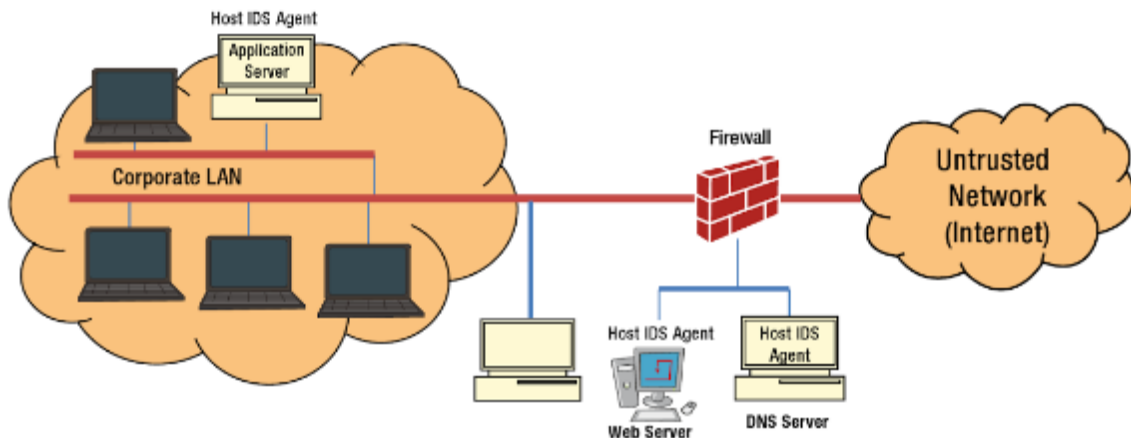


*Figure. Signature based detection*

Signature based detection can offer very specific detection threats by comparing network traffic with the threat signature database. The detection can be enhanced if the network traffic inside the network can be made to learn specific patterns, thus reducing false positives. Signature detection engines tend

### **HOST-BASED IDS:**

Host-based Intrusion Detection System refers to the detection intrusion Single system. This is normally software-based deployment where an agent, as shown in Figure, is installed on the local host that monitors and reports the application activity. HIDS monitors the access to the system and its application and sends alerts for any unusual activities. It **constantly monitors event logs, system logs, application logs**, user policy enforcement, rootkit detection, file integrity and other intrusions to the system. **It constantly monitors these logs and creates a baseline.** If any log entries appear, HIDS Checks the data against the baseline and if entries are found outside of this baseline.



*Figure 11-2. Host-Based Intrusion Detection System*

Most of the HIDS products have ability to prevent attacks also. However, it is initially deployed in the monitor mode and then there is of the System activity, a baseline is and then HIDS is deployed prevention mode. The functionality HIDS depends the logs generated by the System and the fact that the intruders leave evidence of their activities. Generally, hackers get access to the System and install malicious tools so that future access becomes easier.



---

**Advantages of HIDS are:**

- System level protection. Protects from attacks directed to the system
- Any unauthorized activity on the system (configuration changes, file changes, registry changes, etc.) are detected and an alert is generated for further action

**Disadvantages**

- HIDS functionality works only if the Systems generate logs and match against the pre-defined policies. If for some reason, Systems do not generate logs, HIDS may not function properly.
- If hackers bring down the HIDS server, then HIDS is of no use. This is true for any vulnerability Software.

**HOST-BASED IDS/IPS:**

Many host security products contain integrated host-based IDS/IPS systems (HIDS/HIPS), anti-malware and a firewall. These HIDS/HIPS systems have both advantages and weaknesses. They are capable of protecting mobile hosts from an attack when outside the protected internal network, and they can defend local attacks, such as malware in removable devices. They also protect against attacks from network and encrypted attacks in which the encrypted data stream terminates at the host being protected. They have the capability of detecting anomalies of host software execution, e.g., system call patterns. HIDS/HIPS builds a dynamic database of system objects that can be monitored.

On the negative side, if an attacker takes over a host, the HIDS/HIPS and NAC (Network Access Control) agent software can be compromised and disabled, and the audit logs are modified to hide the malware. In addition, HIDS/HIPS has only a local view of the attack, and host -based anomaly detection has a high false alarm rate.