4 th class

Multimedia Security 1

امنية الوسائط المتعددة 1

استاذ المادة : د. منى غازي عبد الصاحب

2023-2024

Lecture 1: Basic Concepts in Data Security

Multimedia communication plays an important role in multiple areas of today's society, including politics, economics, industries, militaries, entertainment, etc. It is of the utmost importance to secure **multimedia** data by providing confidentiality, integrity, and identity or ownership. **Multimedia security** addresses the problems of digital watermarking, data encryption, multimedia authentication, digital rights management, etc. Due to the rapid growth and widespread use of information and communication technologies, Internet services demand better methods of protecting computers, data, and information.

Data security is the protection of programs and data in computers and communication systems against unauthorized modification, destruction, disclosure or transfer whether accidental or intentional.

Data Security Core Principles

The three core principles of data security also referred to as information security are confidentiality, integrity and availability. Below is CIA Triad diagram:

Confidentiality

protecting the information from disclosure to unauthorized parties. It means that sensitive data or information belonging to an organization or government should not be accessed by or disclosed to unauthorized people. Such data include employees' details, classified military information, business financial records etc.

Integrity

Integrity of information refers to protecting information from being modified by unauthorized parties. This means that data should not be modified without owner's authority.

This term covers two related concepts:

Data integrity: Assures that information and programs are changed only in a specified and authorized manner.

System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Availability

Availability of information refers to ensuring that authorized parties are able to access the information when needed. The information must be available on demand. This means that any information system and communication link used to access it must be efficient and functional. An information system may be unavailable due to power outages, hardware failures, unplanned upgrades or repairs.

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

Authenticity

The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the system came from a trusted source.

Accountability

The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports nonrepudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action. Because truly secure systems are not yet an achievable goal, we must be able to trace a security breach to a responsible party. Systems must keep records of their activities to permit later forensic analysis to trace security breaches or to aid in transaction disputes.

Security Architecture

The OSI security architecture is useful to managers as a way of organizing the task of providing security. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined:

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

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

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

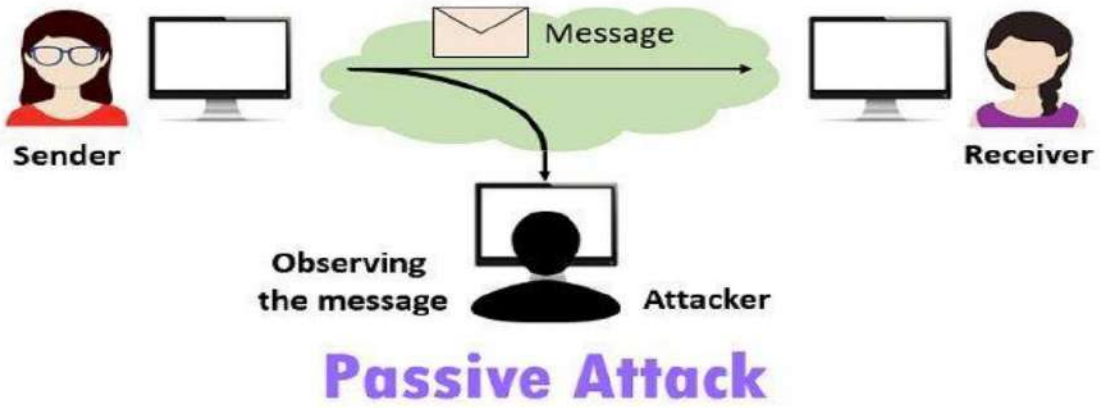
Security Attack:

- any action that compromises the security of information owned by an organization
- information security is about how to prevent attacks, or failing that, to detect attacks on information-based systems
- often threat & attack used to mean same thing
- have a wide range of attacks and can focus on two generic types of attacks: passive and active.

passive attacks: which attempt to gather or make use of information from the system but does not affect system resources. By eavesdropping on, or keeping track of transmissions to:

- obtain message contents or
- monitor traffic flows

Are difficult to detect because they do not involve any alteration of the data. 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: which attempt to alter system resources or affect their operation. By modification of data stream to:

- replay previous messages
- modify messages in transit
- denial of service

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them.

Denial of Service, A "denial-of-service" attack is an attempt by attackers to prevent legal users of a service from using that service. Examples include

- Flooding the network to overloading the servers
- Disrupting connections between systems
- attempts to prevent a particular individual from accessing a service

Security service:

Some services, such as:

1. Authentication

It is a verification of the identity of the user. Some methods of authentication are:

- User ID and passwords. The system compares the given password with a stored password. If the two passwords match, then the user is authentic.
- Swipe card, which has a magnetic strip embedded, which would already contain your details, so that no physical data entry takes place or just a PIN is entered.
- Digital certificate, an encrypted piece of data which contains information about its owner, creator, generation and expiration dates, and other data to uniquely identify a user.

- Biometrics - retinal scanners and fingerprint readers. Parts of the body are considered unique enough to allow authentication to computer systems based on their properties.

2. Access Control

Access control is the security methodology that allows access to information based on identity. Users who have been given permission or keys to information can access it otherwise, access is denied.

3. Encryption-Based Access Control (Privacy) private key

The key used to decode public key messages that must be kept private. A totally different way to control access is to simply encrypt data using public key encryption. Access to the encrypted data is given to those who want it, but it's worthless to them unless they have the private key required to decode it.

Security mechanisms:

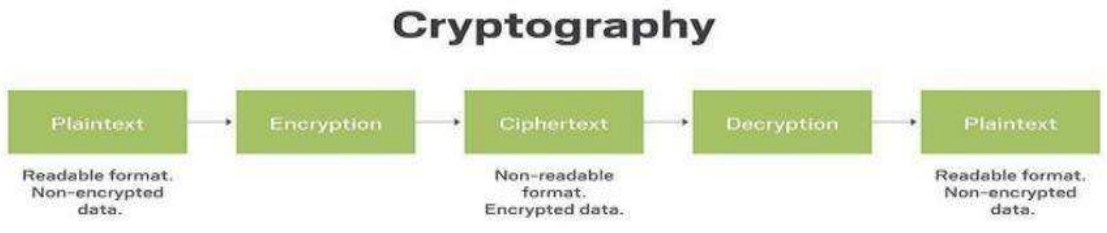
Security mechanisms are technical tools and techniques that are used to implement security services. A mechanism may operate independently or in conjunction with others to offer a certain service.

Cryptographic System

Cryptography deals with all aspects of secure messaging, authentication, digital signatures, and other applications. **Cryptology** is the branch of mathematics that studies the mathematical foundations of cryptographic methods.

In cryptographic terminology, the message is called **plaintext**. The process of encoding the contents of a message in such a way that it hides its contents from outsiders is called **encryption**. The encrypted message is called the **ciphertext**. The process of retrieving the plaintext from the ciphertext is called **decryption**. Encryption and decryption usually make use of a **key**, and the coding method is such that decryption can be performed only by knowing the proper key. **Cryptography** is the art or science of keeping messages secret. **Cryptanalysis** is the art of breaking ciphers, i.e. retrieving the plaintext without knowing the proper key. People who do cryptography are **cryptographers**, and practitioners of cryptanalysis are **cryptanalysts**.

Basic Cryptographic Algorithms: A method of encryption and decryption is called a **cipher**. All modern algorithms use a **key** to control encryption and decryption; a message can be decrypted only if the key matches the encryption key. The key used for decryption can be different from the encryption key, but for most algorithms they are the same.

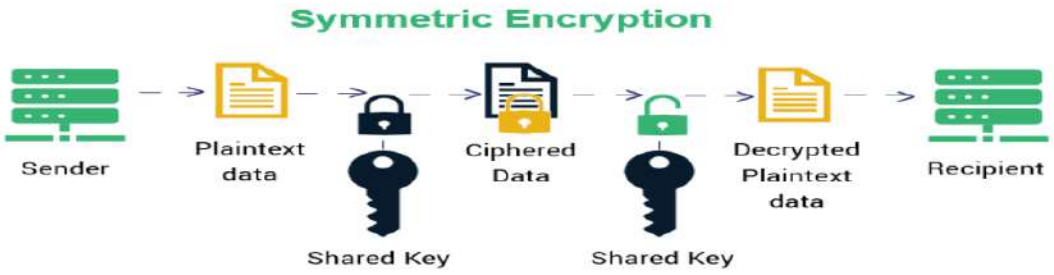


There are two classes of key-based algorithms:

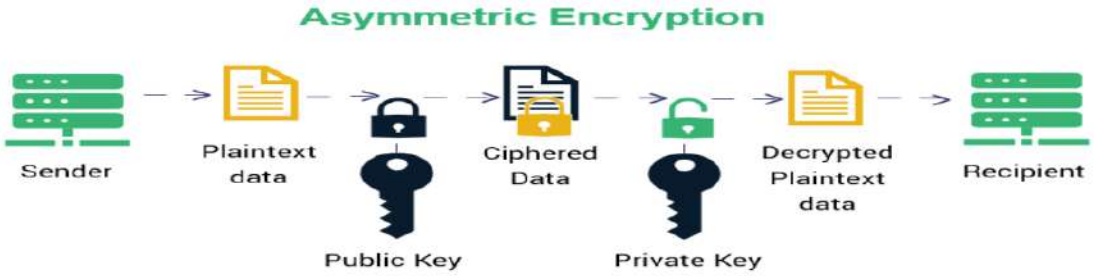
A. **Symmetric** (or **secret-key**) algorithms, use the same key for encryption and decryption (or the decryption key is easily derived from the encryption key), Symmetric algorithms can be divided into **stream ciphers** and **block ciphers**.

   Stream ciphers can encrypt a single bit of plaintext at a time.

   Block ciphers take a number of bits (typically 64 bits in modern ciphers), and encrypt them as a single unit.



B. **Asymmetric** (or **public-key**) algorithms, use a different key for encryption and decryption, and the decryption key cannot be derived from the encryption key. Asymmetric ciphers (also called public-key algorithms or generally public-key cryptography) permit the encryption key to be public, allowing anyone to encrypt with the key, whereas only the proper recipient (who knows the decryption key) can decrypt the message. The encryption key is also called the **public key** and the decryption key the **private key** or **secret key.**



Lecture 2: Mathematical Background

1. Modular Arithmetic

Given any integer a and positive integer n, the division of a by n that leaves the remainder between 0 and n − 1, we define

$$\boldsymbol{a \bmod n}$$

to be the remainder. Note that the remainder must be **between 0 and $\boldsymbol{n - 1}$**, Any integer outside this range is reduced to one in this range by taking the remainder after division by n.

Example: if $a = 13$ and $n = 5$, find $a \ div \ n$, $a \ mod \ n$.

$13 \ div \ 5 = 2$

$13 \ mod \ 5 \ = \ 3$

Example: $11 \ mod \ 7 \ = \ 4$

Example: $-11 \ mod \ 7 \ = \ 3$

Example: $3 \ mod \ 7 = 3$

2. Greatest Common Divisor (GCD)

The Greatest Common Divisor (GCD) of two numbers is the largest possible number which divides both the numbers exactly. The properties of GCD are as given below:

- GCD of two or more numbers divides each of the numbers without a remainder.
- GCD of two or more numbers is a factor of each of the numbers.
- GCD of two or more numbers is always less than or equal to each of the numbers.
- GCD of two or more prime numbers is 1 always.

There are 3 methods to calculate the GCD of two numbers:

1. GCD by listing out the common factors
2. GCD by prime factorization
3. GCD by division method

Each of the above methods is explained in the given solved examples.

Example 1: What is the GCD of 30 and 42?

List the factors of each number.

Factors of 30: 1, 2, 3, 5, 6, 10, 15, 30

Factors of 42: 1, 2, 3, 6, 7, 14, 21, 42

6 is the common factor and the greatest one. The GCD of 30 and 42 is 6.

Example 2: What is the GCD of 60 and 90?

$60 = 2 \times 2 \times 3 \times 5$

$90 = 2 \times 3 \times 3 \times 5$

GCD is the product of the factors that are common to each of the given numbers.

Thus, GCD of 60 and $90 = 2 \times 3 \times 5 = 30$.

Example 3: Find the GCD of 1970 and 1066 using the "division method".

We will divide the larger number by the smaller number. Next, we will make the remainder as the divisor and the last divisor as the dividend and divide again. We will repeat this process until the remainder is 0.

To find gcd(1970, 1066)		
1970	= 1 x 1066 + 904	gcd(1066, 904)
1066	= 1 x 904 + 162	gcd(904, 162)
904	= 5 x 162 + 94	gcd(162, 94)
162	= 1 x 94 + 68	gcd(94, 68)
94	= 1 x 68 + 26	gcd(68, 26)
68	= 2 x 26 + 16	gcd(26, 16)
26	= 1 x 16 + 10	gcd(16, 10)
16	= 1 x 10 + 6	gcd(10, 6)
10	= 1 x 6 + 4	gcd(6, 4)
6	= 1 x 4 + 2	gcd(4, 2)
4	= 2 x 2 + 0	gcd(2, 0)
Therefore, gcd(1970, 1066) = 2		

3. The multiplicative inverse

The multiplicative inverse $a^{-1}\ mod\ n$

$$if\ GCD(a,n) = 1 \quad then \qquad ax\ mod\ n = 1$$

we can find the inverse based on using the equation $a = n \times k + b$

Example 1: Find the multiplicative inverse of 8 mod 11.

11	=	**8**(1)	+	**3**	3	=	11	− 8(1)
8	=	**3**(2)	+	**2**	2	=	8	− 3(2)
3	=	**2**(1)	+	**1**	1	=	3	− 2(1)
2	=	**1**(2)						

Now reverse the process using the equations on the right.

$$1 = 3 - 2(1)$$
$$1 = 3 - (8 - 3(2))(1) = 3 - (8 - (3(2)) = 3(3) - 8$$
$$1 = (11 - 8(1))(3) - 8 = 11(3) - 8(4) = 11(3) + 8(-4)$$

-4+11=**7** the multiplicative inverse of 8 mod 11 is 7

Which satisfy $8 * 7\ mod\ 11 = 1$

Example 2: Find the multiplicative inverse of 11 in Z_{26}.

26=11 *2 + 4 -> b=a- kn = 4=26-11(2)

11=4*2 +3 -> 3=11-4(2)

4= 3*1 +1 -> 1=4-3(1)

3=3*1+ 0

1=4-3(1)

1=4-(11-4(2))(1)=4-11+4(2)=4(3)-11

1=(26-11(2))(3)-11=26(3)-11(6)-11=26(3)-11(7)

So, -7+26=**19** the multiplicative inverse is 19

Which satisfy 11*19 mod 26=1

Example 3: Find the multiplicative inverse of 23 in Z_{100}.

100= 23*4+8 -> 8=100-23(4)

23=8*2+7 -> 7=23-8(2)

8=7*1+1 -> 1=8-7(1)

7=1*7+0

1=8-7(1)=8-(23-8(2))(1)=8-23+8(2)=8(3)-23

(100-23(4)) (3)-23=100(3)-23(12)-23=100(3)-23(13)

So, -13+100=**87** the multiplicative inverse is 87

Which satisfy 23*87 mod 100=1

4. Euler notation

coprime: having no common positive factors other than 1 (also called relatively prime)

Euler's Totient Function: Φ(n) = number of integers less than or equal to n that are coprime with n

1. If prime number $\boldsymbol{\Phi(n) = n - 1}$

Example: Φ (5) = 4 {1,2,3,4}

2. If not prime number $\boldsymbol{\Phi(n^t) = n^{t-1}(n-1)}$

Example: Φ (3^3) = 3^2 (3-1) = 18

3. If n comes from two distinct prime numbers p and q,

$$\boldsymbol{\Phi(n) = \Phi(pq) = \Phi(p) \times \Phi(q) = (p-1) \times (q-1)}$$

Example:

$$\Phi(21) = \Phi(3) \times \Phi(7) = (3-1) \times (7-1) = 12$$

Lecture 3: Types of traditional ciphers systems

Classical Encryption Techniques

There are two classes of key-based algorithms which are symmetric (or secret-key) algorithms and asymmetric (or public-key) algorithms.

Symmetric Cipher Model

Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. Symmetric encryption transforms plaintext into cipher text using a secret key and an encryption algorithm, by using the same key and a decryption algorithm the plaintext is recovered from the ciphertext.

Traditional symmetric ciphers use **transposition** and/or **substitution** techniques.

Transposition techniques systematically transpose the positions of plaintext elements. That means a mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols.

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.

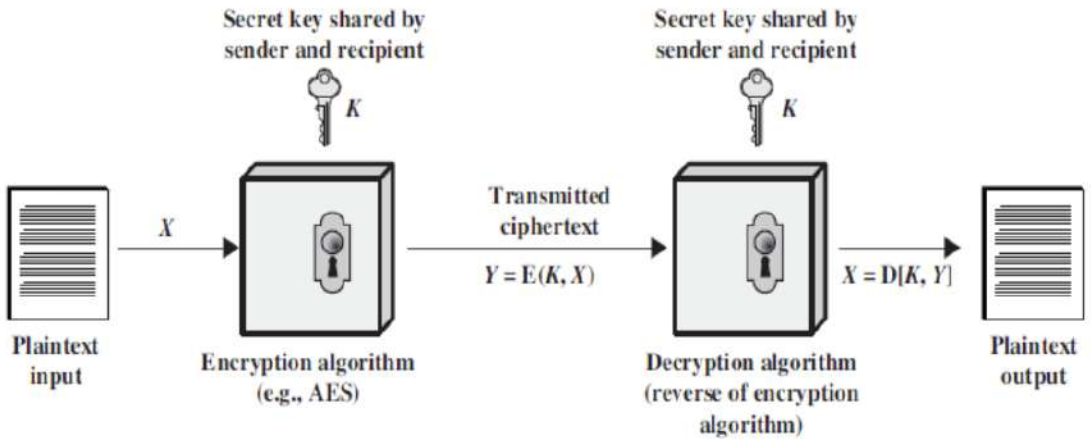
• **Ciphertext**: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

• **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of conventional encryption:

1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key.

2. Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.



Transposition Ciphers systems

Transposition Ciphers are a method of encryption where the positions of the plaintext (which are commonly characters or groups of characters) are shifted according to a regular system to produce the ciphertext. Following are some methods of transposition ciphers:

1. Rail Fence cipher

2. Columnar transposition

1. Rail Fence Cipher

Write message letters out diagonally over a number of rows, then read off cipher row by row.

Example: Encrypt the message "**Meet me after the toga party**" with a rail fence of depth 2:

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

e t e f e t e o a a t

Giving ciphertext: **mematrhtgpryetefeteoaat**

2. Columnar Transposition

In a columnar transposition, the message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword. For example, the keyword ZEBRAS is of length 6 (so the rows are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

Example: Suppose we use the keyword "**ZEBRAS"** and the message "**WE ARE DISCOVERED FLEE AT ONCE"** In a **regular** columnar transposition

6	**3**	**2**	**4**	**1**	**5**
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	F	L	E
E	A	T	O	N	C
E	*Q*	*K*	*J*	*E*	*U*

providing five nulls (QKJEU), these letters can be randomly selected as they just fill out the incomplete columns and are not part of the message.
The ciphertext is then read off as: **EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE**

In the **irregular** case, the columns are not completed by nulls:

6	**3**	**2**	**4**	**1**	**5**
W	E	A	R	E	D
I	S	C	O	V	E
R	E	D	F	L	E
E	A	T	O	N	C
E					

This results in the ciphertext: **EVLN ACDT ESEA ROFO DEEC WIREE**

Example: Key: **3 4 2 1 5 6 7** , Plaintext: **“attack postponed until two am”**
Plaintext:

3	**4**	**2**	**1**	**5**	**6**	**7**
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: aptmttnaaodwtsuocoixknlypetz

Substitution Ciphers systems

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. There are several types of substitution ciphers, including monoalphabetic, homophonic, polyalphabetic, and polygram substitution ciphers.

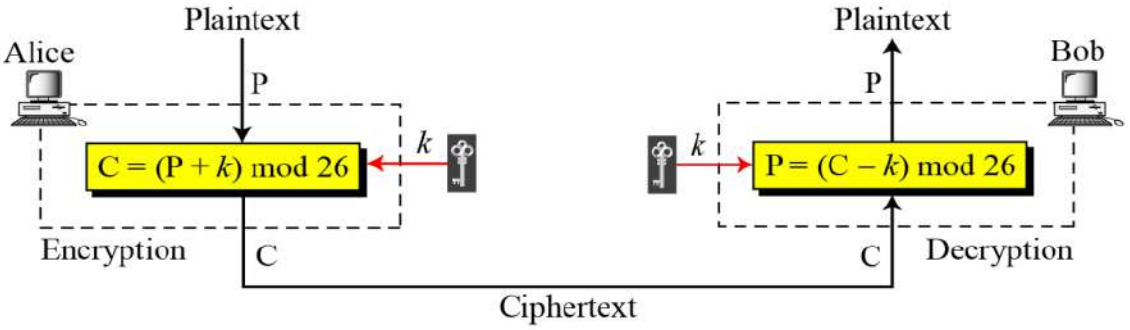
1. Monoalphabetic Ciphers

In monoalphabetic substitution, the relationship between a symbol in the plaintext to a symbol in the ciphertext is always **one-to-one**.

1.1 Additive Cipher, Caesar cipher

The Additive cipher is the simplest monoalphabetic cipher and is sometimes called a **Shift cipher** and **Caesar cipher.**

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



When the cipher is additive, the plaintext, ciphertext, and key are integers in $\mathbf{Z_{26}}$.

The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. Note that the alphabet is wrapped around, so that the letter following Z is A.

Example: Use the additive cipher with key = 15 to encrypt the message "hello".

Solution: We apply the **encryption** algorithm to the plaintext, character by character:

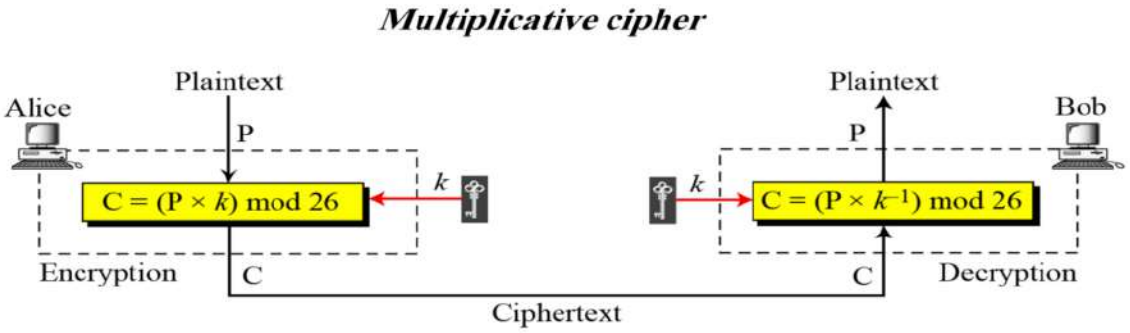
Plaintext: h → 07	Encryption: (07 + 15) mod 26	Ciphertext: 22 → W
Plaintext: e → 04	Encryption: (04 + 15) mod 26	Ciphertext: 19 → T
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: l → 11	Encryption: (11 + 15) mod 26	Ciphertext: 00 → A
Plaintext: o → 14	Encryption: (14 + 15) mod 26	Ciphertext: 03 → D

We apply the **decryption** algorithm to the ciphertext character by character:

Ciphertext: W → 22	Decryption: (22 – 15) mod 26	Plaintext: 07 → h
Ciphertext: T → 19	Decryption: (19 – 15) mod 26	Plaintext: 04 → e
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: A → 00	Decryption: (00 – 15) mod 26	Plaintext: 11 → l
Ciphertext: D → 03	Decryption: (03 – 15) mod 26	Plaintext: 14 → o

1.2 Multiplicative Ciphers

In a multiplicative cipher, the plaintext and ciphertext are integers in Z_{26}; the key is an integer in Z_{26}*.



The key domain for any multiplicative cipher needs to be in Z26*. This set has only **12 members**: **1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25**.

Number	1	3	5	7	9	11	15	17	19	21	23	25
Multiplicative inverse	1	9	21	15	3	19	7	23	11	5	17	25

$1 \times 1 = 1 \bmod 26$, $3 \times 9 = 26 = 1 \bmod 26$, $5 \times 21 = 105 = 1 \bmod 26$, $7 \times 15 = 105 = 1 \bmod 26$, $11 \times 19 = 209 = 1 \bmod 26$, $17 \times 23 = 391 = 1 \bmod 26$, and $25 \times 25 = 625 = 1 \bmod 26$. But, 2, for example, does not have an inverse; there is no number mod 26 that 2 can be multiplied by that will result in 1.

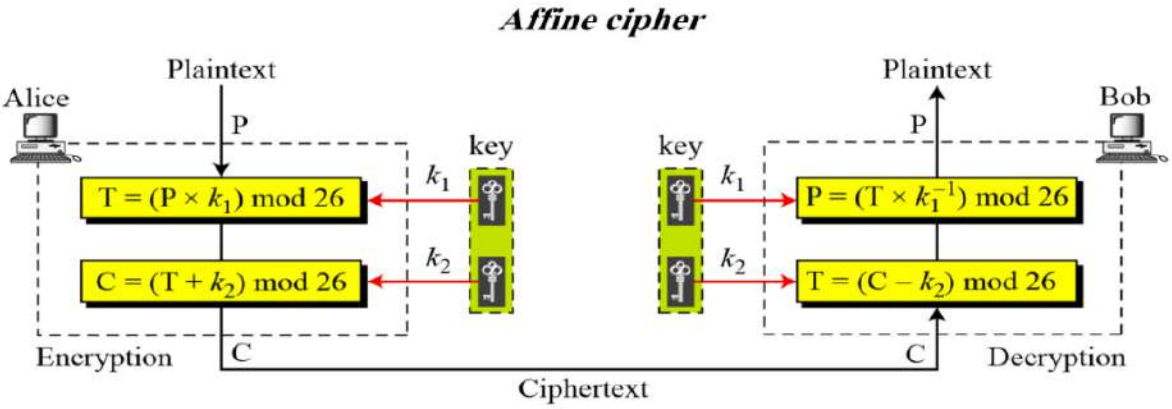
Multiplication modulo 26

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2	2	4	6	8	10	12	14	16	18	20	22	24	26	2	4	6	8	10	12	14	16	18	20	22	24	26
3	3	6	9	12	15	18	21	24	1	4	7	10	13	16	19	22	25	2	5	8	11	14	17	20	23	26
4	4	8	12	165	20	24	2	6	10	14	18	22	26	4	8	12	16	20	24	2	6	10	14	18	22	26
5	5	10	15	20	25	4	9	14	19	24	3	8	13	18	23	2	7	12	17	22	1	6	11	16	21	26
6	6	12	18	24	4	10	16	22	2	8	14	20	26	6	12	18	24	4	10	16	22	2	8	14	20	26
7	7	14	21	2	9	16	23	4	11	18	25	6	13	20	1	8	15	22	3	10	17	24	5	12	19	26
8	8	16	24	6	14	22	4	12	20	2	10	18	26	8	16	24	6	14	22	4	12	20	2	10	18	26
9	9	18	1	10	19	2	11	20	3	12	21	4	13	22	5	14	23	6	15	24	7	16	25	8	17	26
10	10	20	4	14	24	8	18	2	12	22	6	16	26	10	20	4	14	24	8	18	2	12	22	6	16	26
11	11	22	7	18	3	14	25	10	21	6	17	2	13	24	9	20	5	16	1	12	23	8	19	4	15	26
12	12	24	10	22	8	20	6	18	4	16	2	14	26	12	23	10	22	8	20	6	18	4	16	2	14	26
13	13	26	13	26	13	26	13	26	13	26	13	26	13	26	13	26	13	26	13	26	13	26	13	26	13	26
14	14	2	16	4	18	6	20	8	22	10	24	12	26	14	2	26	4	18	6	20	8	22	10	24	12	26
15	15	4	19	8	23	12	1	16	5	20	9	24	13	2	17	6	21	10	25	14	3	18	7	22	11	26
16	16	6	22	12	2	18	8	24	14	4	20	10	26	16	6	22	12	2	18	8	24	14	4	20	10	26
17	17	8	25	16	7	24	15	6	23	14	5	22	13	4	21	12	3	20	11	2	19	10	1	18	9	26
18	18	10	2	20	12	4	22	14	6	24	16	8	26	18	10	2	20	12	4	22	14	6	24	16	8	26
19	19	12	5	24	17	10	3	22	15	8	1	20	13	6	25	18	11	4	23	16	9	2	21	14	7	26
20	20	14	8	2	22	16	10	4	24	18	12	6	26	20	14	8	2	22	16	10	4	24	18	12	6	26
21	21	16	11	6	1	22	17	12	7	2	23	18	13	8	3	24	19	14	9	4	25	20	15	10	5	26
22	22	18	14	10	6	2	24	20	16	12	8	4	26	22	18	14	10	6	2	24	20	16	12	8	4	26
23	23	20	17	14	11	8	5	2	25	22	19	16	13	10	7	4	1	24	21	18	15	12	9	6	3	26
24	24	22	20	18	16	14	12	10	8	6	4	2	26	24	22	20	18	16	14	12	10	8	6	4	2	26
25	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	26
26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26	26

Example: We use a multiplicative cipher to encrypt the message “hello” with a key of 7. The ciphertext is “XCZZU”.

Plaintext: h $\rightarrow$ 07	Encryption: $(07 \times 07) \bmod 26$	ciphertext: 23 $\rightarrow$ X
Plaintext: e $\rightarrow$ 04	Encryption: $(04 \times 07) \bmod 26$	ciphertext: 02 $\rightarrow$ C
Plaintext: l $\rightarrow$ 11	Encryption: $(11 \times 07) \bmod 26$	ciphertext: 25 $\rightarrow$ Z
Plaintext: l $\rightarrow$ 11	Encryption: $(11 \times 07) \bmod 26$	ciphertext: 25 $\rightarrow$ Z
Plaintext: o $\rightarrow$ 14	Encryption: $(14 \times 07) \bmod 26$	ciphertext: 20 $\rightarrow$ U

1.3 Affine Ciphers



$$C = (P \times k_1 + k_2) \bmod 26 \qquad P = ((C - k_2) \times k_1^{-1}) \bmod 26$$

where k_1^{-1} is the multiplicative inverse of k_1 and $-k_2$ is the additive inverse of k_2

The affine cipher uses a pair of keys in which the first key is from Z_{26}^* and the second is from Z_{26}. The size of the key domain is $26 \times 12 = 312$.

Example: Use an affine cipher to encrypt the message "hello" with the key pair (7, 2).

P: h → 07	Encryption: $(07 \times 7 + 2) \bmod 26$	C: 25 → Z
P: e → 04	Encryption: $(04 \times 7 + 2) \bmod 26$	C: 04 → E
P: l → 11	Encryption: $(11 \times 7 + 2) \bmod 26$	C: 01 → B
P: l → 11	Encryption: $(11 \times 7 + 2) \bmod 26$	C: 01 → B
P: o → 14	Encryption: $(14 \times 7 + 2) \bmod 26$	C: 22 → W

Example: Use the affine cipher to decrypt the message "ZEBBW" with the key pair (7, 2) in modulus 26.

C: Z → 25	Decryption: $((25 - 2) \times 7^{-1}) \bmod 26$	P:07 → h
C: E → 04	Decryption: $((04 - 2) \times 7^{-1}) \bmod 26$	P:04 → e
C: B → 01	Decryption: $((01 - 2) \times 7^{-1}) \bmod 26$	P:11 → l
C: B → 01	Decryption: $((01 - 2) \times 7^{-1}) \bmod 26$	P:11 → l
C: W → 22	Decryption: $((22 - 2) \times 7^{-1}) \bmod 26$	P:14 → o

Lecture 4:

2. Homophonic Substitution

Homophonic Substitution is a simple way to make monoalphabetic substitution more secure, by levelling out the frequencies with which the ciphertext letters appear. It was an early attempt to make Frequency Analysis a less powerful method of cryptanalysis. The basic idea behind homophonic substitution is to allocate more than one letter or symbol to the higher frequency letters. For example, you might use 6 different symbols to represent "e" and "t", 2 symbols for "m" and 1 symbol for "z". We need to use a key of some form to order the letters of the ciphertext alphabet, we use the letters from the keyword first, without repeats, then use the rest of the alpha-numeric alphabet. However, we assign multiple spaces to some letters. Using the keyphrase "18 fresh tomatoes and 29 cucumbers".

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	1	8	F	R	E	S	H	T	O	M	A	N	D	2	9	C	U	B	G	I	J	K	L	P	Q	V	W	X	Y	Z	0	3	4	5	6	7

Example: encrypt the message "run away, the enemy are coming" using the keyphrase above.

Encryption: To generate the ciphertext alphabet, we replace each letter in the plaintext with one of the options in the ciphertext alphabet for that letter.

the ciphertext "Q0I 1486, YNH OGSB6 1QH RKB2GA".

Decryption: To decrypt the message "4O 8QH E2WRJ3SQTE" we have to simply look for each ciphertext letter along the bottom row then the plaintext "we are discovered".

3. Polyalphabetic Ciphers

Because additive, multiplicative, and affine ciphers have small key domains, they are very vulnerable to **brute-force attack**. A better solution is to create a mapping between each plaintext character and the corresponding ciphertext character. In polyalphabetic substitution, each occurrence of a character may have a different substitute. The relationship between a character in the plaintext to a character in the ciphertext is **one-to-many**.

3.1 Vigenère Cipher

The best known, and one of the simplest polyalphabetic substitutions, such algorithm is referred to as the Vigenère cipher. To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. The process of encryption is simple: Given a key letter x and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled x and the column labeled y; in this case the ciphertext is V. Decryption is equally simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column. The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword.

$P = P_1\ P_2\ P_3 \ldots\ C = C_1\ C_2\ C_3 \ldots\ K = [(k_1, k_2, ..., k_m), (k_1, k_2, ..., k_m), \ldots]$

Encryption: $C_i = P_i + k_i$ **Decryption: $P_i = C_i - k_i$**

we can say that the additive cipher is a special case of Vigenere cipher in which m = 1.

A Vigenere Tableau

	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	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

Example: We can encrypt the message "**She is listening**" using the 6-character keyword "**PASCAL**".

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***
C's values:	**07**	**07**	**22**	**10**	**18**	**22**	**23**	**18**	**11**	**6**	**13**	**19**	**02**	**06**
Ciphertext:	**H**	**H**	**W**	**K**	**S**	**W**	**X**	**S**	**L**	**G**	**N**	**T**	**C**	**G**

4. Polygram Substitution Cipher

Polygram cipher systems are ciphers in which group of letters are encrypted together, and includes enciphering large blocks of letters. Therefore, permits arbitrary substitution for groups of characters. For example, the plaintext group "ABC" could be encrypted to "RTQ", "ABB" could be encrypted to "SLL", and so on. In another meaning, encryption includes substitution of a block of multiple letters from plaintext with the corresponding group of ciphertext. Example of such ciphers are **Playfair**, and **Hill ciphers**.

4.1 Playfair Cipher

playfair cipher is a diagram substitution cipher, the key is given by a **5*5 matrix** of 25 letters (j was not used), as described in the below figure.
Each pair of plaintext letters are encrypted according to the following rules:
l. if m_1 and m_2 are in the same row, then c_1 and c_2 are to the right of m_1 and m_2, respectively. The first column is considered to the right of the last column.
2. if m_1 and m_2 are in the same column, then c_1 and c_2 are below m_1 and m_2 respectively. the first row is considered to be below the last row.
3. if m_1 and m_2 are in different rows and columns, then c_1 and c_2 are the other two corners of the rectangle.
4. If $m_1=m_2$ a null letter is inserted into the plaintext between m_1 and m_2 to eliminate the double.
5. If the plaintext has an odd number of characters, a null letter is appended to the end of the plaintext.

Example:

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

Figure of Key for Playfair cipher

M = RE NA IS SA NC EX

Ek(M) = HG WC BH HR WF GV

4.2 Hill Cipher

Hill cipher performs linear transformation on plaintext characters to get cipher text characters.

Key in the Hill cipher

$$K = \begin{bmatrix} k_{11} & k_{12} & \cdots & k_{1m} \\ k_{21} & k_{22} & \cdots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{m1} & k_{m2} & \cdots & k_{mm} \end{bmatrix}$$

$$\begin{aligned} C_1 &= P_1 k_{11} + P_2 k_{21} + \cdots + P_m k_{m1} \\ C_2 &= P_1 k_{12} + P_2 k_{22} + \cdots + P_m k_{m2} \\ &\cdots \\ C_m &= P_1 k_{1m} + P_2 k_{2m} + \cdots + P_m k_{mm} \end{aligned}$$

The key matrix in the Hill cipher needs to have a multiplicative inverse.

If d = 2, M= m_1 m_2, then C = Ek(M) = C_1 C_2 where:

$C_1 = (k_{11}\, m_1 + k_{12}\, m_2) \bmod n$

$C_2 = (k_{21}\, m_1 + k_{22}\, m_2) \bmod n$

Expressing M and C as column vectors:

C = Ek(M) = KM where K is matrix of coefficients:

$$\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \quad \text{that is} \quad \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \bmod n$$

Deciphering is done using the inverse matrix K^{-1}

Dk =(C)= K^{-1} C mod n = K^{-1} K M mod n =M

Where K K^{-1} mod n = I, I is 2*2 identity matrix.

Where k * k^{-1} mod 26 = $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

If a square matrix A has a nonzero determinant, then the inverse of the matrix is computed as $[A^1]_{ij} = (1)^{i+j} (D_{ij})/\text{ded}(A)$, where (D_{ij}) is the subdeterminant formed by deleting the ith row and the jth column of A and det(A) is the determinant of A. For our purposes, all arithmetic is done mod 26.

Example: The plaintext "code is ready" can make a 3 × 4 matrix when adding extra bogus character "z" to the last block and removing the spaces. The ciphertext is "OHKNIHGKLISS".

Example

$$\underset{C}{\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}} = \underset{P}{\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}} \underset{K}{\begin{bmatrix} 09 & 07 & 11 & 13 \\ 04 & 07 & 05 & 06 \\ 02 & 21 & 14 & 09 \\ 03 & 23 & 21 & 08 \end{bmatrix}}$$

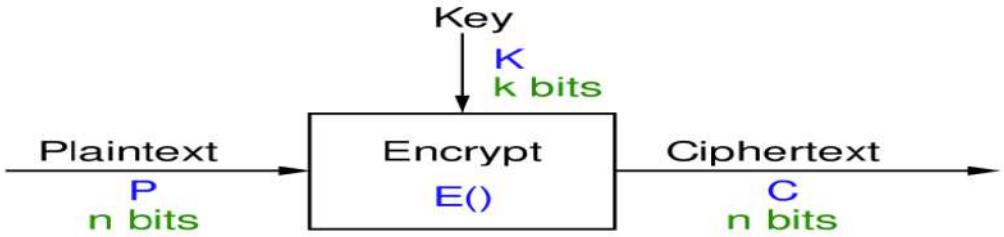
a. Encryption

$$\underset{P}{\begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix}} = \underset{C}{\begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix}} \underset{K^{-1}}{\begin{bmatrix} 02 & 15 & 22 & 03 \\ 15 & 00 & 19 & 03 \\ 09 & 09 & 03 & 11 \\ 17 & 00 & 04 & 07 \end{bmatrix}}$$

b. Decryption

Lecture 5: Block Cipher and DES

Block Cipher: a block cipher is a method of encrypting data in blocks to produce ciphertext using a cryptographic key and algorithm. The block cipher processes fixed-size blocks simultaneously, as opposed to a stream cipher, which encrypts data one bit at a time. Most modern block ciphers are designed to encrypt data in fixed-size blocks of either 64 or 128 bits. In a modern block cipher (but still using a classical encryption method), A block cipher operates on a block of N bits from the plaintext to produce a block of N bits of the ciphertext.



Feistel Cipher - An iterate block 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. 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.

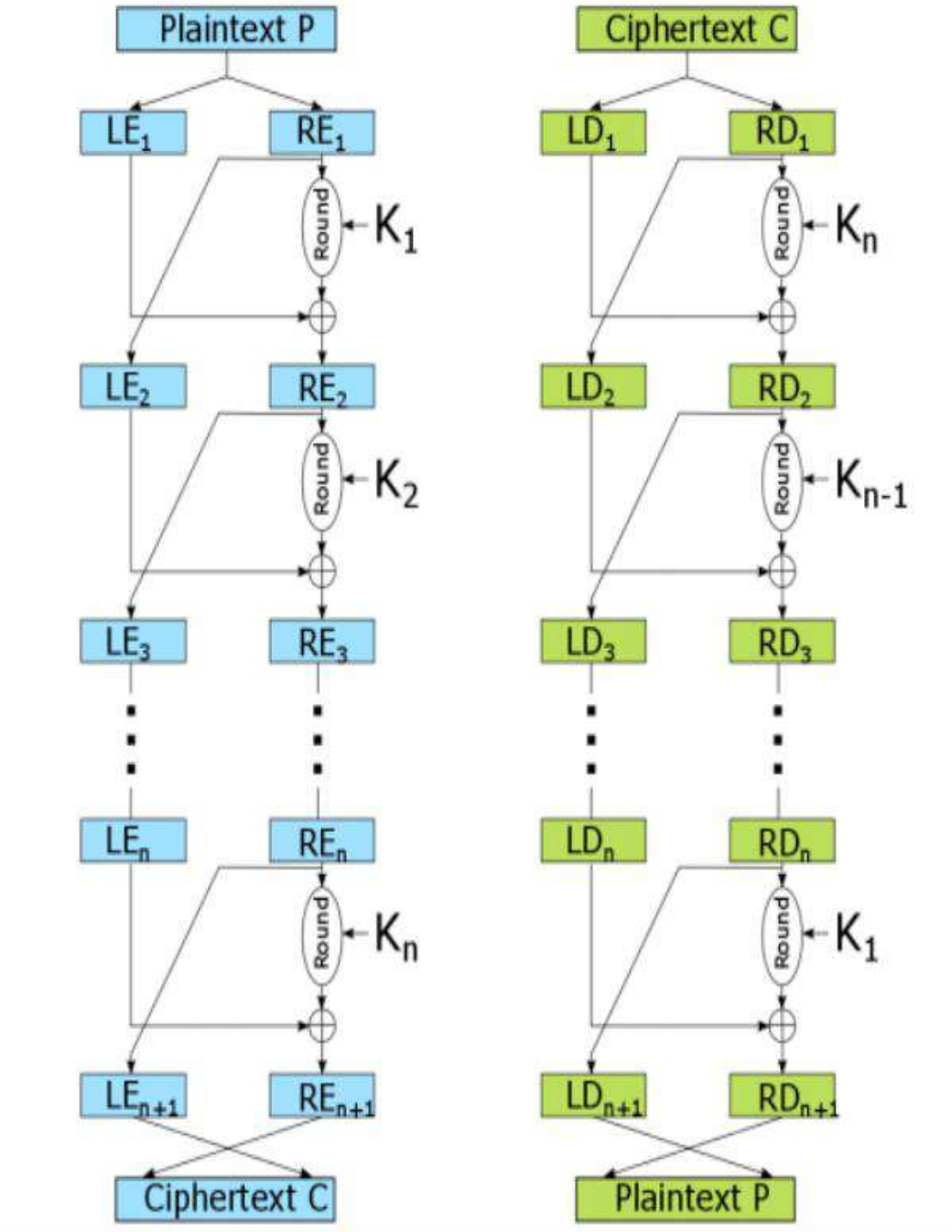


Figure: Feistel Structure.

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. The mechanism of **diffusion** seeks to make the statistical relationship between the plaintext and ciphertext as complex as possible.

• **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. The mechanism of **confusion** seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible.

• **Number of rounds:** The single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

• **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.

Data Encryption Standard (DES)

Data Encryption Standard (DES) Cipher Algorithm - A 16-round Feistel cipher with block size of 64 bits. DES stands for Data Encryption Standard. The Data Encryption Standard (DES), known as the Data Encryption Algorithm (DEA) by ANSI and the DEA-1 by the ISO, has been most widely used block cipher in world, especially in financial industry. It encrypts 64-bit data, and uses 56-bit key with 16 sub-keys of 48-bit.

Description of DES

DES is a block cipher; it encrypts data in 64-bit blocks. A 64-bit block of plaintext goes in one end of the algorithm and a 64-bit block of ciphertext comes out the other end. DES is a symmetric algorithm: The same algorithm and key are used for both encryption and decryption.

The key length is 56 bits. (The key is usually expressed as a 64-bit number, but every eight bit is used for parity checking and is ignored. These parity bits are the least- significant bits of the key bytes.) The key can be any 56-bit number and can be changed at any time. All security rests within the key.

At its simplest level, the algorithm is nothing more than a combination of the two basic techniques of encryption: confusion and diffusion. The fundamental building block of DES is a single combination of these techniques (a substitution followed by a permutation) on the text, based on the key. This is known as a round. DES has 16 rounds; it applies the same combination of techniques on the plaintext block 16 Times, as shown in the following Figure.

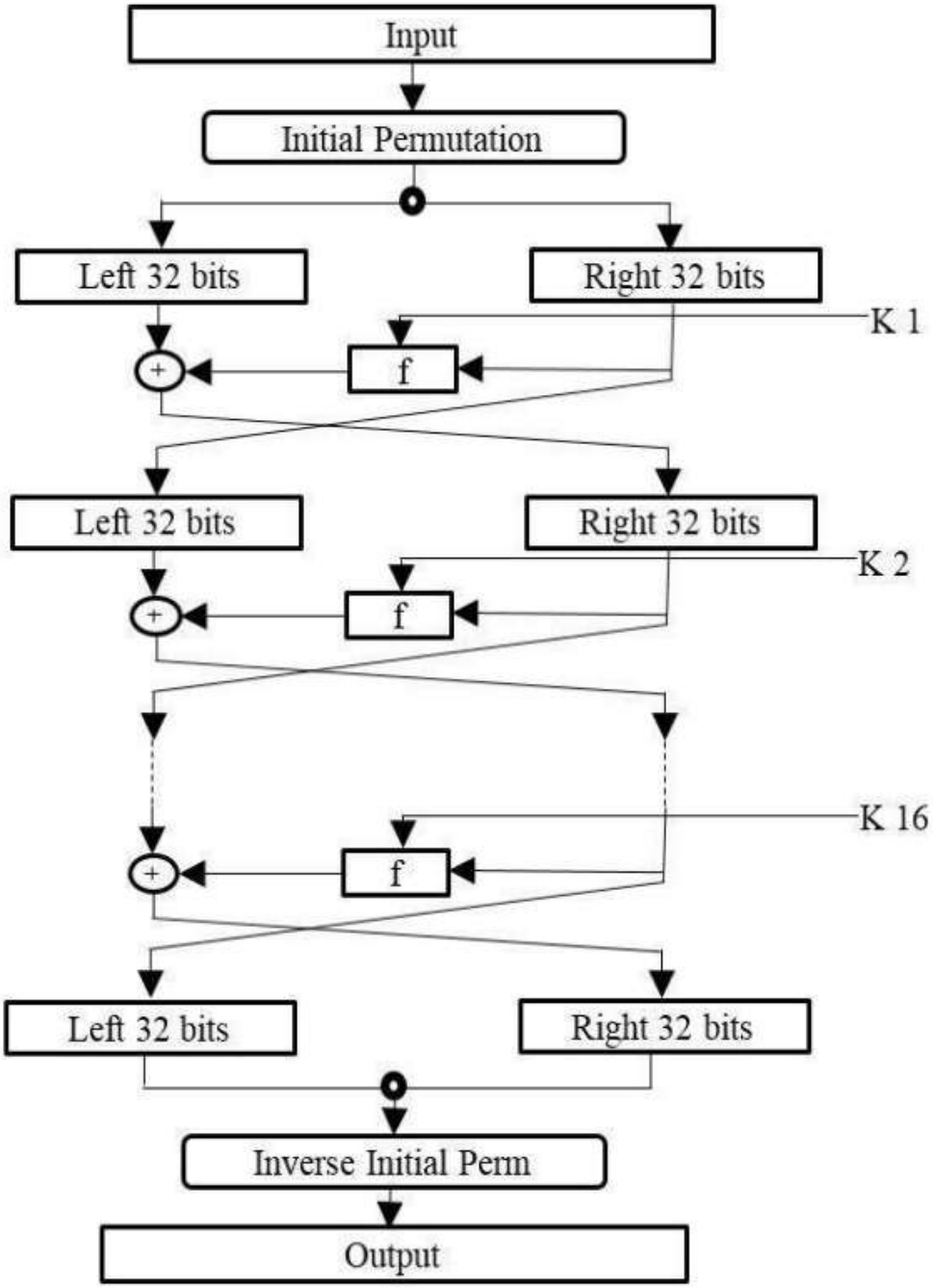


Figure: DES Structure.

The DES Algorithm

The basic process in enciphering a 64-bit data block using the DES consists of:

- an initial permutation (IP)
- 16 rounds of a complex key dependent calculation f
- final permutation, being the inverse of IP

In each round, the key bits are shifted, and then 48 bits are selected from the 56 bits of the key. The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again. These four operations make up Function f. The output of Function f is then combined with the left half via another XOR. The result of these operations becomes the new right half; the old right half becomes the new left half.

If Bi is the result of the ith iteration, Li and Ri are the left and right halves of Bi, Ki is the 48-bit key for round i, and f is the function that does all the substituting and permuting and XORing with the key, then a round looks like:

$$\boldsymbol{Li = R\,i-1}$$

$$\boldsymbol{Ri = L\,i-1\;XOR\;f\;(Ri-1, Ki)}$$

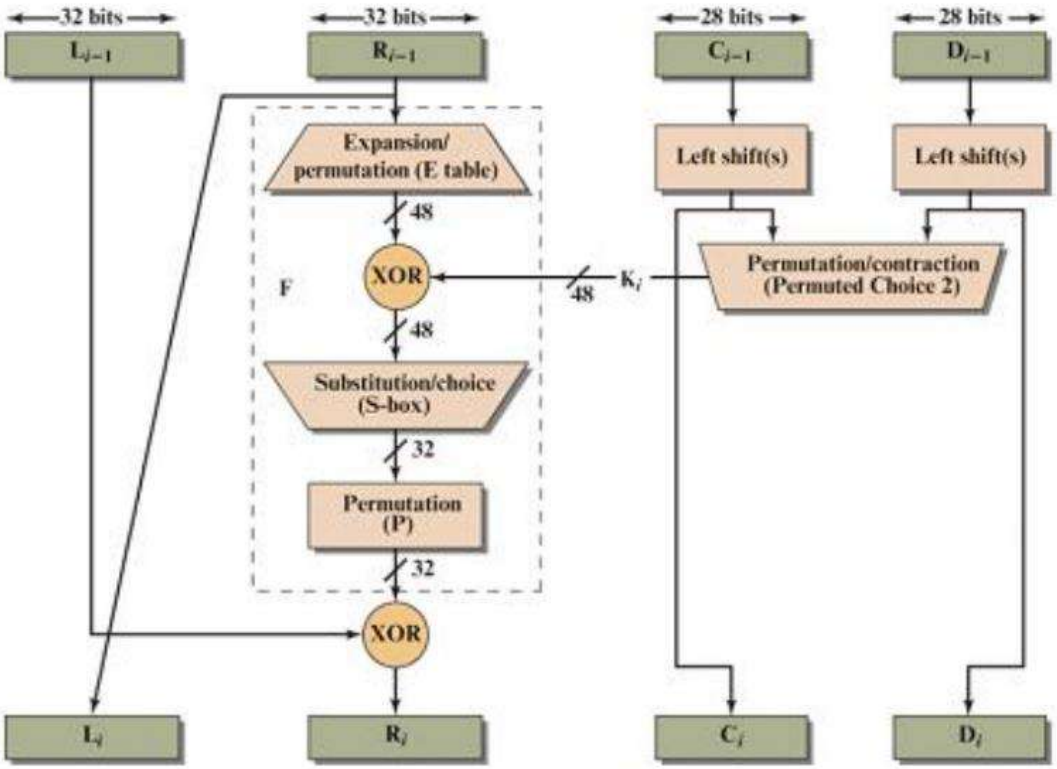


Figure: One Round Of DES

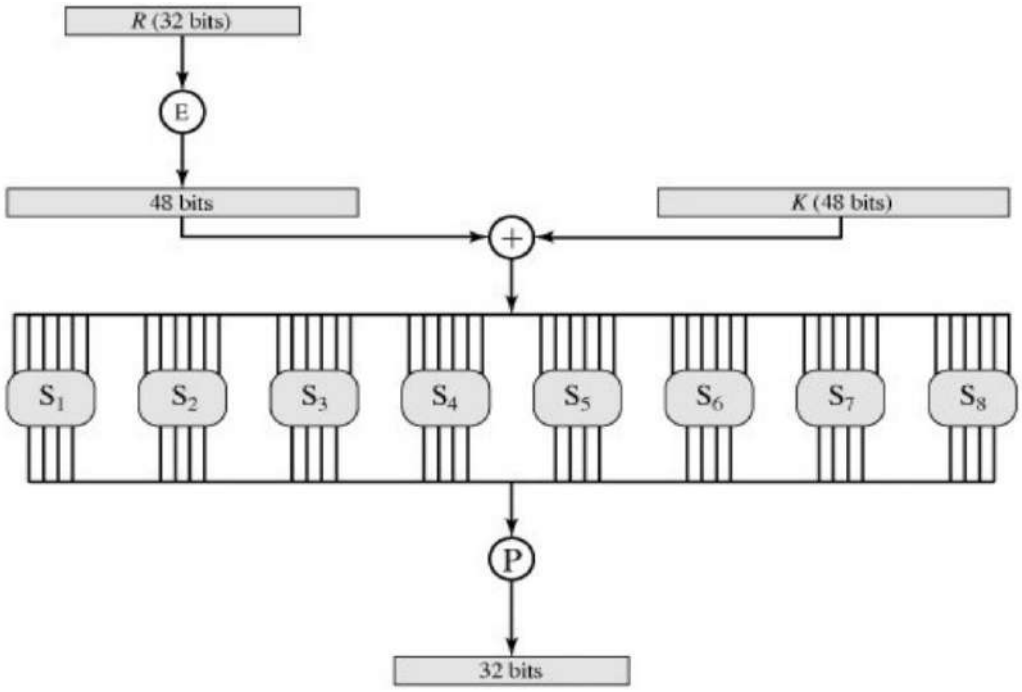
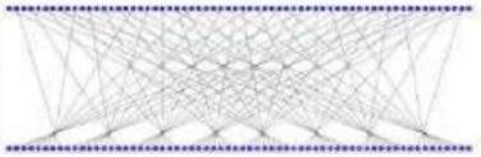


Figure: F-function and S-boxes

1. Initial Permutation

The initial permutation occurs before round 1; it transposes the input block as described in Table. For example, the initial permutation moves bit 58 of the plaintext to bit position 1, bit 50 to bit position 2, bit 42 to bit position 3, and so forth. The initial permutation and the corresponding final permutation do not improve DES's security, just make DES more complex.

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



2. Key Transformation

Initially, the 64-bit DES key is reduced to a 56-bit key by ignoring every eighth bit. Let us call this operation PC1.

PC2 is the operation which reduces the 56-bits key to a 48-bits subkey for each of the 16 rounds of DES. These subkeys, Ki, are determined in the following manner. PC1 splits the key bits into 2 halves (C and D), each 28-bits. The halves C and D are circularly shifted left by either one or two bits, depending on the round. This shift is given in Table 1. After being shifted, 48 out of the 56 bits are selected. This is done by an operation called compression permutation, it permutes the order of the bits as well as selects a subset of bits. Table 2 defines the compression permutation.

Table 1: Number of Key Bits Shifted per Round

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 2: Compression Permutation

14, 17, 11, 24, 1, 5, 3, 28, 15, 6, 21, 10,
23, 19, 12, 4, 26, 8, 16, 7, 27, 20, 13, 2,
41, 52, 31, 37, 47, 55, 30, 40, 51, 45, 33, 48,
44, 49, 39, 56, 34, 53, 46, 42, 50, 36, 29, 32.

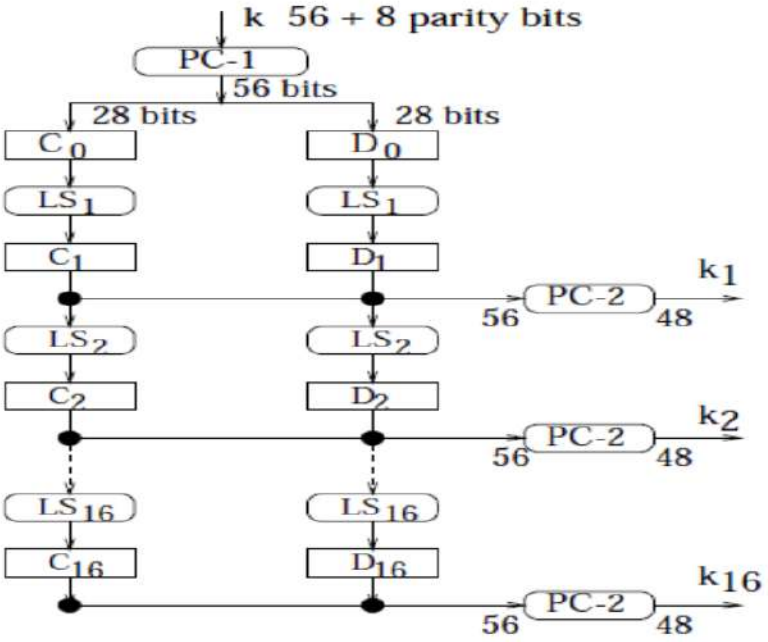


Figure: 16th key generation

3. Expansion Permutation

This operation expands the right half of the data, Ri, from 32 bits to 48 bits. Because this operation changes the order of the bits as well as repeating certain bits, it is known as an expansion permutation. This operation has two purposes: It makes the right half the same size as the key for the XOR operation and it provides a longer result that can be compressed during the substitution operation. However, neither of those is its main cryptographic purpose. Table below shows which output positions correspond to which input

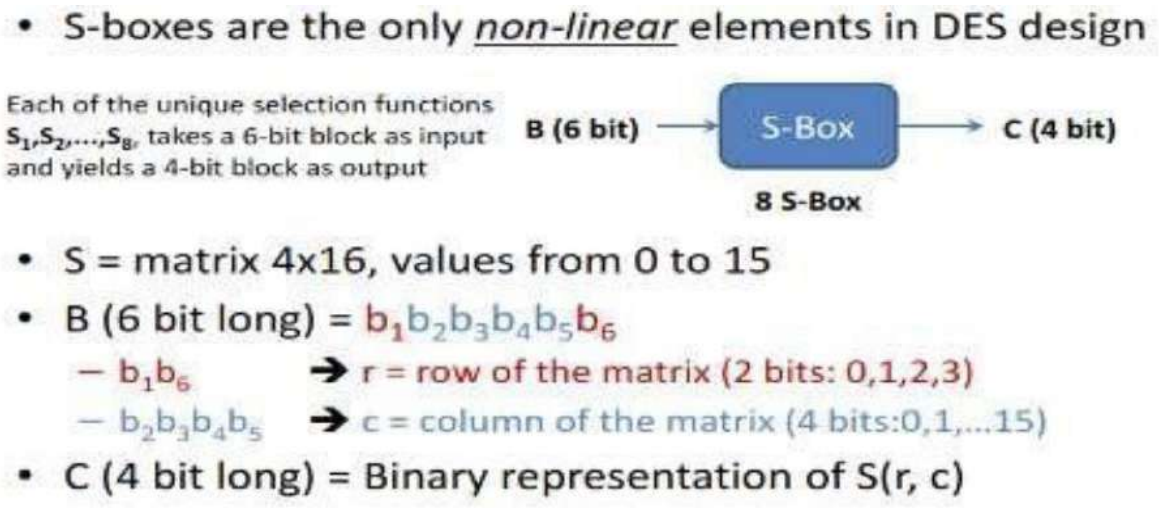
positions. For example, the bit in position 3 of the input block moves to position 4 of the output blocks.

Expansion Permutation Table

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.

4. S-Box Substitution

After the compressed key is XORed with the expanded block, the 48-bit result moves to a substitution operation. The substitutions are performed by eight substitution boxes, or S-boxes. Each S-box has a 6-bit input and a 4-bit output, and there are eight different S-boxes. The 48 bits are divided into eight 6-bit sub-blocks. Each separate block is operated on by a separate S-box: The first block is operated on by S-box 1, the second block is operated on by S-box 2, and so on. The result of this substitution phase is eight 4-bit blocks which are recombined into a single 32-bit block. This block moves to the next step: the P-box permutation.



The DES S-Boxes are the following:

S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

5. P-Box Permutation

The 32-bit output of the S-box substitution is permuted according to a P-box. This permutation maps each input bit to an output position; no bits are used twice and no bits are ignored. The Table below shows the position to which each bit moves.

P-Box Permutation Table

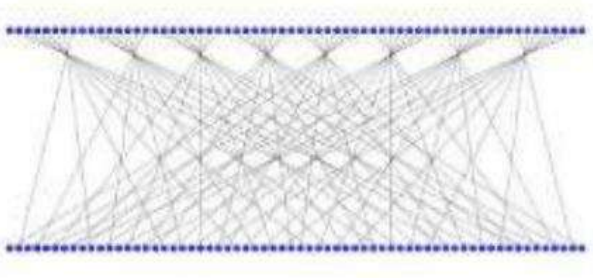
16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26, 5, 18, 31, 10, 2, 8, 24, 14, 32, 27, 3, 9, 19, 13, 30, 6, 22, 11, 4, 25

Finally, the result of the P-box permutation is XORed with the left half of the initial 64-bit block. Then the left and right halves are switched and another round begins.

6. Final Permutation

The final permutation is the inverse of the initial permutation. Note that the left and right halves are not exchanged after the last round of DES; instead the concatenated block R_{16} L_{16} is used as the input to the final permutation. There's nothing going on here; exchanging the halves and shifting around the permutation would yield exactly the same result. This is so that the algorithm can be used to both encrypt and decrypt.

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



Decrypting DES

After all the substitutions, permutations, XORs, and shifting around, you might think that the decryption algorithm is completely different and just as confusing as the encryption algorithm. The same algorithm works for both encryption and decryption. With DES it is possible to use the same function to encrypt or decrypt a block. The only difference is that the keys must be used in the reverse order. That is, if the encryption keys for each round are K1, K2, K3, . . . , K16, then the decryption keys are K16, K15, K14, . . . , K1,. The algorithm that generates the key used for each round is circular as well. The key shift is a right shift and the number of positions shifted is 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

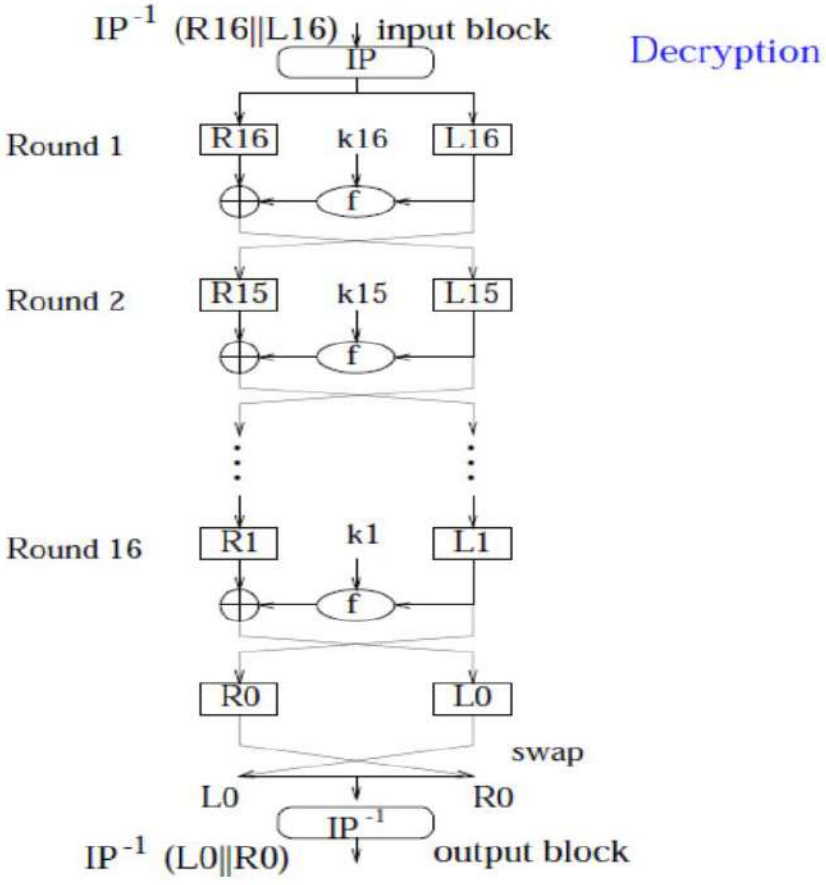


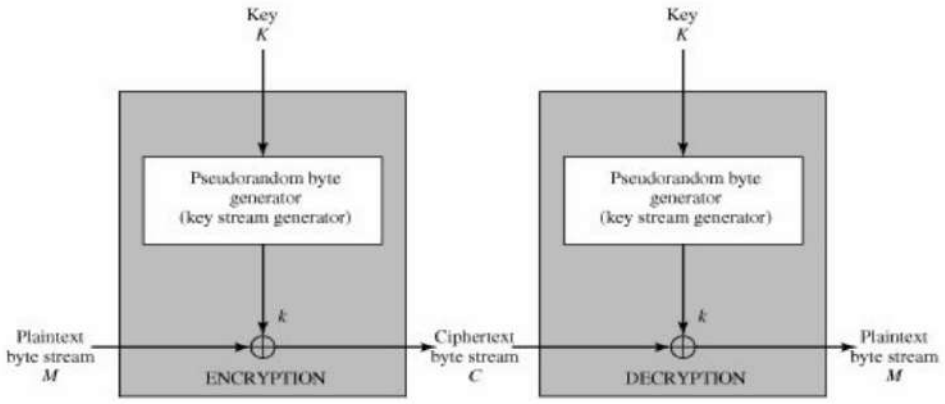
Figure: DES Decryption.

Lecture 6: Stream Cipher and Randomness Tests

Stream Cipher

A typical stream cipher encrypts plaintext one byte at a time; although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. The following figure represents the diagram of stream cipher structure. In this structure a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. For now, we simply say that a pseudorandom stream is one that is unpredictable without knowledge of the input key. The output of the generator, called a key stream, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation. For example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is

```
  11001100  plaintext
⊕ 01101100  key stream
  10100000  ciphertext
```



Decryption requires the use of the same pseudorandom sequence:

```
  10100000  ciphertext
⊕ 01101100  key stream
  11001100  plaintext
```

Important considerations for designing a stream cipher:

1. The encryption sequence should have a large period. A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats. The longer the period of repeat the more difficult it will be to do cryptanalysis. This is essentially the same consideration that was discussed with reference to the Vigenère cipher, namely that the longer the keyword the more difficult the cryptanalysis.

2. The key stream should approximate the properties of a true random number stream as close as possible. For example, there should be an approximately equal number of 1s and 0s. If the key stream is treated as a stream of bytes, then all of the 256 possible byte values should appear approximately equally often. The more random appearing the key stream is, the more randomized the ciphertext is, making cryptanalysis more difficult.

3.The output of the pseudorandom number generator is conditioned on the value of the input key. To guard against brute-force attacks, the key needs to be sufficiently long.

Note: Pseudorandom Number Generators (PRNGs)

Cryptographic applications typically make use of algorithmic techniques for random number generation. These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random. However, if the algorithm is good, the resulting sequences will pass many reasonable tests of randomness. Such numbers are referred to as pseudorandom numbers.

One-Time Pad or Vernam Cipher

• The one-time pad, which is a provably secure cryptosystem, was developed by Gilbert Vernam.

• The message is represented as a binary string (a sequence of 0's and 1's using a coding mechanism such as ASCII coding.

• The key is a truly random sequence of 0's and 1's of the same length as the Message, used only once, and kept secret between the sender and receiver for the encryption to be secure.

• The encryption is done by adding the key to the message modulo 2, bit by bit. This process is often called exclusive or, and is denoted by XOR. The symbol ⊕ is used.

Example: Let the message be "IF" then its ASCII code be (1001001 1000110) and the key be (1010110 0110001). The ciphertext can be found exoring message and key bits as the following:

Encryption:

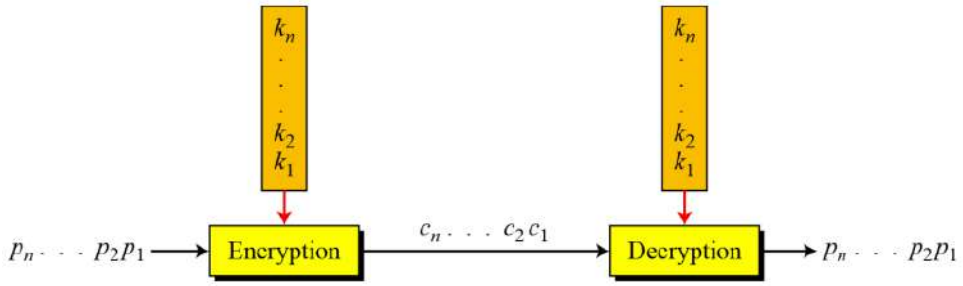
1001001 1000110 **Plaintext**
1010110 0110001 **Key**

0011111 1110111 **Ciphertext**

Decryption:

0011111 1110111 **Ciphertext**
1010110 0110001 **Key**

1001001 1000110 **Plaintext**



Basic Idea comes from One-Time-Pad cipher,

$$\begin{aligned}
\text{Encryption} \quad &: c_i = m_i \oplus k_i \qquad i = 1,2,3,\ldots \\
m_i \quad &: \text{plain-text bits.} \\
k_i \quad &: \text{key (key-stream) bits} \\
c_i \quad &: \text{cipher-text bits.} \\
\text{Decryption} \quad &: m_i = c_i \oplus k_i \qquad i = 1,2,3,\ldots
\end{aligned}$$

Drawback: Key-stream should be as long as plain-text. Key distribution & Management difficult.

Solution: Stream Ciphers (in which key-stream is generated in pseudo-random fashion from relatively short secret key.

Randomness: Closely related to unpredictability.

Randomness Tests

Pseudorandom number generators require tests as exclusive verifications for their "randomness," as they are decidedly not produced by "truly random" processes, but rather by deterministic algorithms. If a given sequence was able to pass all of these tests within a given degree of significance, then it was judged to be random.

1. Frequency Test: Used to ensure that there is roughly the same number of O's and 1's using the following formula:

$$\mathbf{X^2 = (n0 - n1)^2 / n}$$

$\mathbf{X^2 = 0}$ **if** $\mathbf{n0 = n1}$ For good sequence $\mathbf{0 < X^2 < 3.84}$

2. Serial Test: The serial test, did the same thing but for sequences of two digits at a time. Used to ensure that the transition probabilities are reasonable. This will give as some level of confidence that each bit is independent of its predecessor suppose .01 occurs n01, 10 occurs n10, 00 occurs n00…and 11 occurs n11 times.

$$X^2 = 4/n - 1\ \Sigma(nij)^2 - 2/n\ \Sigma(ni)^2 + 1 \qquad \text{For good sequence } X^2 <= 5.99.$$

3. Poker Test: The poker test, tested for certain sequences at a time. For any binary sequence of length n , there are 2^m different possibilities. In this test we partition our sequence into blocks of size m, and then we count the frequency of each type of sections of length m in the sequence. If the frequencies are f_0, f_1,.. f_{m-1} We compute:

$$X^2 = 2^m /F\ \Sigma((Xi)^2 / mi)\ - F$$

Where $F = \Sigma\ fi = n/m$

4. Auto Correlation Test: For a given sequence of n bits S1,..Sn then

A (d) = ∑ai ai+d 0 <= d <= n-1

A (0) =∑ai = ni

If the sequence has n0 of zeros and n1 of ones then

$$\mu = n1^2\ (n-d) / n^2 \text{ The test will pass if } \quad X^2 = (A(d)- \mu)^2 /\mu$$

$$X^2 <= 3.841 \text{ for all values of } d$$

5. Runs Test: Divide the sequence into blocks and gaps, let r0i be the number of gaps of length i and r1i be the number of blocks of length i. If r0 and r1 are number of gaps and blocks respectively then: r0 = ∑r0i and r1 = ∑ r1i This test applied after the sequence had passed the serial test, then we would expect about 1/2 the gaps with length 1, 1/4 of gaps of length 2, and so on. Ideally $1/2^i$ of runs having length i.

Example: Given the following sequence of minimal length of **31**.

0101011101100011111001101001000

Sol.

1. Frequency test

n0=15 , n1=31-15=16

$X^2= (15-16)^2/ n = (-1)^2/31 = 1/31 = 0.0322$ $0.0322 < 3.841$ Pass

2.Serial test:

n00=6 , n01=8 , n10=8 , n11=8

$X^2= 4/n-1 (n00^2+ n01^2+ n10^2+ n11^2) - 2/n (n0^2+n1^2) + 1$

$= 4/30 (36 + 64 + 64 + 64) - 2/31(15^2+16^2) + 1$

$=30.4 - 31.032 + 1 = 0.368$ $0.368 < 5.99$ Pass

3.Poker test

$n=31, n=2^5$ then m=5

F=n/m =31/5= 6 Then we have 6 blocks of 5 digit each

01010 11101 10001 11110 01101 00100 0

We count Xi where i is number of ones in each block

X0=0, X1=1, X2=2, X3=1, X4=2, X5=0

$X^2= 2^m/F \Sigma((Xi)^2/ mi) – F$ Where $F = \Sigma fi = n/m$

$X^2 = 2^5/6 \sum(xi)^2/5i - 6$

where xi = x0, x1, x2, x3, x4, x5= 0,1,2,1,2,0 WHERE i = 0 ,1,2,3,4,5

$X^2 = 5.33 (0^2/5*0 + 1^2/5*1 + 2^2/5*2 + ….) - 6$

$= 5.33(0 + 1/5 + 4/10 + 1/10 + 4/5 + 0) - 6$

$=5.33(2 + 4 + 1 + 8/10) – 6$

$=5.33*15/10 - 6 =1.995$

Checking x^2table with degree of freedom of 2^2- 1 = 31 then the test is pass

4. Auto correlation test

n0= 15 n1=16

$\mu = (16)^2(31-l) / (31)^2=7.99$ for d=1

$\mu = 7.72$ for d=2

$\mu = 7.45$ for d=3

01010111011000111111001101001000

10101110110001111100110100100

A (1) =8

$X^2 = (8 - 7.99)^2/ 7.99 = 0.0000125$

01010111011000111111001101001000

01011101100011111001101001000

A (2) =8

$X^2= (8-7.72)^2/ 7.72= 0.01$

01010111011000111111001101001000

10111011000111111001101001000

A (3) =8

$X^2 = (8-7.45)^2/ 7.45 = 0.04$ And so on $X^2 <= 3.84$ for all value of d

5.runs test

r0=9, r1=8, n01=8= r0-1, n10=8= r1

n11 =n1 –r1 = 16 - 8 = 8

n00 = 6= n0-r0 = 15-9 = 6

r01=5 r11=4

r02=2 r12=2

r03=2 r13=1

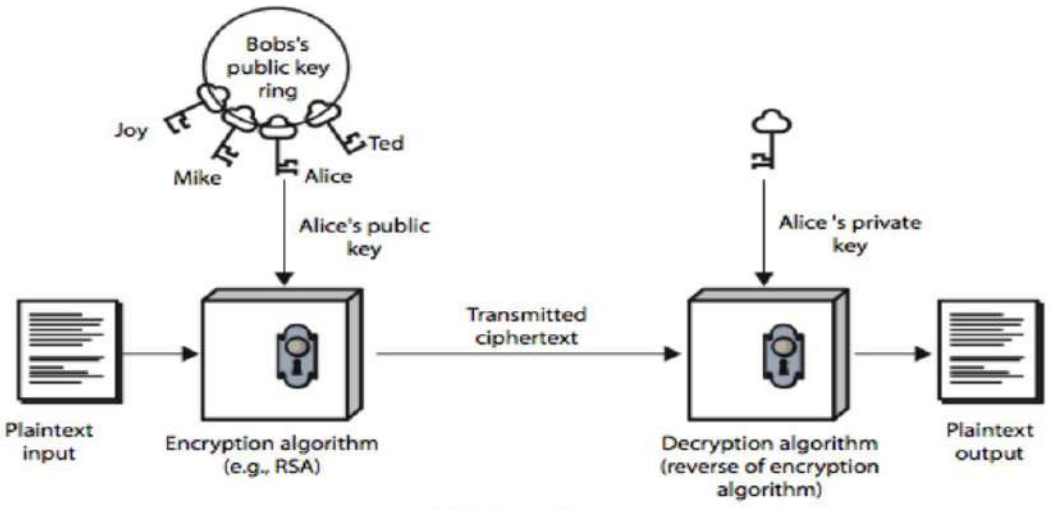
r14=0

r15=1

Lecture 7: Public-Key Cryptography and RSA

Public-Key Cryptography

- 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.
- **public-key/two-key/asymmetric** cryptography involves the use of two keys:
  1. a **public-key**, which may be known by anybody, and can be used to encrypt messages, and verify signatures.
  2. a **private-key**, known only to the recipient, used to decrypt messages, and sign (create) signatures.
- is asymmetric because
  - those who encrypt messages or verify signatures cannot decrypt messages or create signatures.



(a) Encryption

Public-Key Characteristics

- it is computationally infeasible to find decryption key knowing only algorithm & encryption key
- it is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known

Public-Key Applications

- can classify uses into 3 categories:
  - Encryption /decryption: The sender encrypts a message with the recipient's public key.
  - Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
  - Key exchange: Two sides cooperate to exchange a session key. encryption/decryption (provide secrecy)
- some algorithms are suitable for all uses, others are specific to one.

RSA Description and Algorithm

RSA stands for Rivest, Shamir, and Adleman, they are the inventors of the RSA cryptosystem. RSA is one of the algorithms used in PKI (Public Key Infrastructure), **asymmetric key** encryption scheme. RSA is a **block chiper**, it encrypts message in blocks (block by block). The common size for the key length now is 1024 bits for P and Q, therefore N is 2048 bits, if the implementation (the library) of RSA is fast enough, we can double the key size.

RSA Key Generation Algorithm:

1. Generate two large random primes, $\boldsymbol{p}$ and $\boldsymbol{q}$, of approximately equal size such that their product $\boldsymbol{n = pq}$ is of the required bit length, e.g., 1024 bits.
2. Compute $\boldsymbol{n = pq}$ and $\boldsymbol{\varphi(n) = (p-1)(q-1)}$.
3. Choose an integer $\boldsymbol{e}$, $\boldsymbol{1 < e < \varphi(n)}$, such that $\boldsymbol{gcd(e, \varphi(n)) = 1}$.
4. Compute the secret exponent $\boldsymbol{d}$, such that $\boldsymbol{d = e^{-1} \bmod \varphi(n)}$.
5. The **public key is** $\boldsymbol{(n, e)}$ and the **private key is** $\boldsymbol{(n, d)}$. Keep all the values d, p, q and $\varphi(n)$ secret.

- $\boldsymbol{n}$ is known as the modulus.
- $\boldsymbol{e}$ is known as the public exponent or encryption exponent or just the exponent.
- $\boldsymbol{d}$ is known as the secret exponent or decryption exponent.

RSA Encryption:

Sender A does the following: -

1. Obtains the recipient B's public key (n, e).
2. Represents the plaintext message as a positive integer m.
3. Computes the ciphertext $\boldsymbol{c = m^e \bmod n}$.
4. Sends the ciphertext c to B.

RSA Decryption:

Recipient B does the following: -

1. Uses his private key (n, d) to compute $\boldsymbol{m = c^d \bmod n}$.
2. Extracts the plaintext from the message representative m.

Example 1:

1. Select primes p=11, q=3.
2. n = pq = 11*3 = 33

$\varphi(n)$ = (p-1)(q-1) = 10*2 = 20

3. Choose e=3

gcd (e, $\varphi(n)$) = gcd(3, 20) = 1

4. Compute d such that ed ≡ 1 (mod $\varphi(n)$)

i.e. compute d = e^{-1} mod $\varphi(n)$ = 3^{-1} mod 20

i.e. find a value for d such that $\varphi(n)$ divides (ed-1)

i.e. find d such that 20 divides 3d-1.

Simple testing (d = 1, 2, ...) gives d = 7

Check: ed-1 = 3*7 - 1 = 20, which is divisible by $\varphi(n)$.

5. Public key = (n, e) = (33, 3)

Private key = (n, d) = (33, 7).

Now say we want to encrypt the message m = 7,

$c = m^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33 = 13.$

Hence the ciphertext c = 13.

To check decryption we compute

$m' = c^d \bmod n = 13^7 \bmod 33 = 7.$

We can break down a potentially large number into its components and combine the results of easier. **a = bc mod n = (b mod n).(c mod n) mod n**

example: $m' = 13^7 \bmod 33 = 13^{(3+3+1)} \bmod 33 = 13^3.13^3.13 \bmod 33$

$= (13^3 \bmod 33).(13^3 \bmod 33).(13 \bmod 33) \bmod 33$

$= (2197 \bmod 33).(2197 \bmod 33).(13 \bmod 33) \bmod 33$

$= 19.19.13 \bmod 33 = 4693 \bmod 33 = 7.$

Example 2:

Select p: 47 , q: 71

n = p*q = 3337

$\varphi(n)$ = (p-1)*(q-1) = 3220

Guess a large value for public key e then we can work down from there.

enter trial public key e: 79

Use private key d: 1019

Publish e: 79

and n: 3337

cipher = plain ^e (mod n) ---------- plain = cipher^d (mod n)

Example 3:

1) Generate two large prime numbers, p and q To make the example easy to follow I am going to use small numbers, but this is not secure.

p = 7 , q = 19

2) Let n = pq ---------- n = 7 * 19 = 133

3) Let PHi $\varphi(n)$ = (p - 1)(q - 1)

$\varphi(n)$ = (7 - 1)(19 - 1) = 6 * 18 = 108

4) Choose a small number, e coprime to $\varphi(n)$.

e coprime to PHi, means that the largest number that can exactly divide both e and m (their greatest common divisor, or GCD) is 1.

e = 2 => GCD(e, 108) = 2 (no)

e = 3 => GCD(e, 108) = 3 (no)

e = 4 => GCD(e, 108) = 4 (no)

e = 5 => GCD(e, 108) = 1 (yes!)

5) Find d

This is equivalent to finding d which satisfies de = 1 +x $\varphi(n)$ where x is any integer. We can rewrite this as **d = (1 + x $\boldsymbol{\varphi(n)}$) / e**. Now we work through values of x until an integer solution for e is found:

x = 0 => d = 1 / 5 (no)

x = 1 => d = 109 / 5 (no)

x = 2 => d = 217 / 5 (no)

x = 3 => d = 325 / 5 = 65 (yes!)

Public Key n = 133 && e = 5

Secret Key n = 133 && d =65

Encryption

The message must be a number less than the smaller of p and q. However, at this point we don't know p or q, so in practice a lower bound on p and q must be published. This can be somewhat below their true value and so isn't a major security concern. For this example, let’s use the message "6".

C = P^ e mod n

= 6^ 5 mod 133

= 7776 mod 133 = 62

Decryption

$P = C^d \bmod n$

$= 62^{65} \bmod 133$

$= 62 * 62^{64} \bmod 133$

$= 62 * (62^2)^{32} \bmod 133$

$= 62 * 3844^{32} \bmod 133$

$= 62 * (3844 \bmod 133)^{32} \bmod 133$

$= 62 * 120^{32} \bmod 133$

$= 62 * 36^{16} \bmod 133 = 62 * 99^8 \bmod 133$

$= 62 * 92^4 \bmod 133 = 62 * 85^2 \bmod 133$

$= 62 * 43 \bmod 133 = 2666 \bmod 133 = 6$

Lecture 8: Digital Signatures based on RSA

Digital signatures can be created by reversing the roles of encryption and decryption with message recovery.

Secrecy and Authenticity

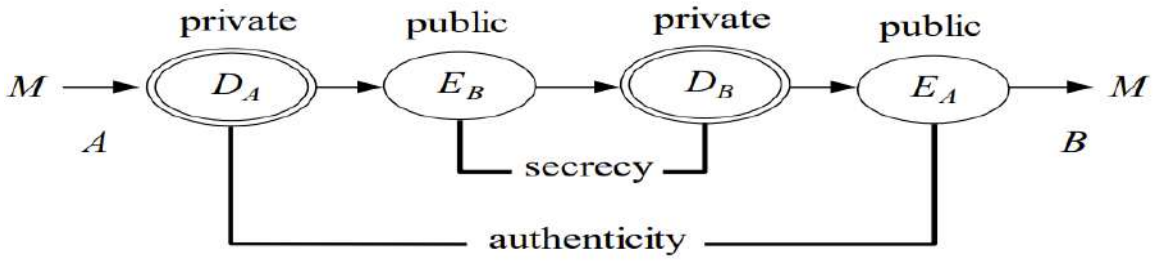
In a public-key system, secrecy and authenticity are both provided.

For Secrecy Suppose user A wishes to send a message M to another user B. If A knows B's public transformation E_B, A can transmit M to B in secrecy by sending the ciphertext $C = E_B(M)$.

On receipt, B deciphers C using B's private transformation D_B, getting $D_B(C) = D_B(E_B(M)) = M$

For authenticity, M must be transformed by A's own private transformation D_A. A sends $C = D_A(M)$ to B.

On receipt uses A's public transformation EA to compute $E_A(C)= E_A(D_A(M)) =M$.



Key Generation

1. Generate two large, distinct primes p, q
2. Compute $n = p \times q$ and $\varphi(n) = (p-1) \times (q-1)$
3. Select a random number $1 < e < \varphi(n)$ such that $\gcd(e, \varphi(n)) = 1$
4. Compute the unique integer $1 < d < \varphi(n)$ such that $ed \equiv 1 \bmod \varphi(n)$
5. (d, n) is the private key
6. (e, n) is the public key

Signature Generation and Verification

- **Signature generation:** In order to sign a message $\boldsymbol{m}$, A does the following:

1. Compute $\boldsymbol{s} = \mathbf{m}^{\mathbf{d}} \boldsymbol{mod}\ \boldsymbol{n}$; then A's signature for m is $\boldsymbol{s}$
2. Encrypted Message-Signature Pair.

- **Signature verification:** In order to verify A's signature s and recover message m, B does the following:

1. Obtain A's authentic public key (e, n).
2. Decoding the message.
3. checks to see if the signature **s** is valid.

Example: Alice and Bob want to communicate via RSA and digitally sign their messages so that they cannot be forged by an evil third party.as the following:

RSA setup

- **Bob:** selects the primes $p_B = 5$, $q_B = 13$, and computes $n_B = p_B * q_B = 65$, $\varphi(n_B) = (p_B - 1)(q_B - 1) = 48$.
- He chooses $e_B = 5$, relatively prime to 48, and computes $\mathbf{d_B = 29}$ (the multiplicative inverse of 5 modulo 48).
- **Bob's Public key (65,5), and the other information private.**
- **Alice:** $p_A = 3$, $q_A = 11$; $n_A = p_A * q_A = 33$; $\varphi(n_A) = (p_A - 1)(q_A - 1) = 20$; $e_A = 3$, $\mathbf{d_A = 7}$.
- **Alice's public key (33,3), and keeps the rest a secret.**

Message-signature pair

- Bob wants to send Alice the message **m = 9**.
- First, he digitally **signs** the message using **his private** RSA key.

$S = m^{d}_{B} \bmod n_B = 9^{29} \bmod 65 = 3^{58} \bmod 65 = 3^{10} \bmod 65 = 29 \bmod 65 = 29$

- The digital signature of the message is **S = 29**.

Encrypted message-signature pair

- Bob **encrypts** the message signature pair (m,S) using **Alice's public** modulus n_A and encryption key e_A.

$y = m^{e}_{A} \bmod n_A = 9^3 \bmod 33 = 3$

$R = S^{e}_{A} \bmod n_A = 29^3 \bmod 33 = 2$

- The encrypted message signature pair is **(y, R) = (3, 2)**.

Decoding the message

- Alice receives the pair (y, R) = (3, 2) (from Bob). She **decrypts** the message using **her private** decryption key $\mathbf{d_A = 7}$.

$m = y^{d}_{A} \bmod n_A = 3^7 \bmod 33 = 9$

$S = R^{d}_{A} \bmod n_A = 2^7 \bmod 33 = 29$

The decrypted message-signature pair is **(m,S) = (9,29).**

- Alice **checks** to see if the **signature S is valid**. She looks up **Bob's public** RSA, which has modulus $n_B = 65$ and encryption key $e_B = 5$.

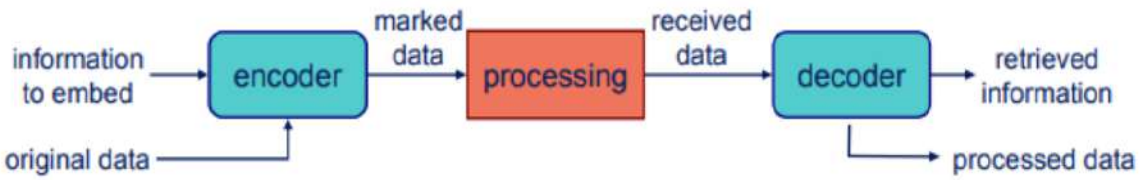
So, she computes $z = S^{e}_{B} \bmod n_B = 29^5 \bmod 65 = 9$.

She compares z with m, **z = m = 9**, **so the digital signature is valid and the message is authentic**.

Lecture 9: Information Hiding

Information Hiding

Information Hiding: Communication of information by embedding it in and retrieving it from other digital data. Depending on application we may need process to be robust and secure, etc. we use hide data Because you want to protect it from malicious use, copy protection (Digital Watermarks), or because you do not want anyone to even know about its existence Covert communication (Steganography)



Where can we hide?

- Media
  - Video
  - Audio
  - Still Images
  - Documents
- Software
- Hardware designs

The Needed to Data Hiding

• Covert communication using images.

• Ownership of digital images, authentication, copyright

• Adding captions to images, additional information.

Steganography

Steganography is the art and science of invisible communication. This is accomplished through hiding information in other information. The word steganography is derived from the Greek words steganos (meaning hidden or covered) and the Greek root graph (meaning to write) defining it as “hidden writing”.

Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination. The use of steganography can be combined with encryption as an extra step for hiding or protecting data.

Steganography can be used to hide almost any type of digital content, including text, image, video or audio content; the data to be hidden can be hidden inside almost any other type of digital content. The content to be concealed through steganography -- called hidden text -- is often encrypted before being incorporated into the cover data stream. If not encrypted, the hidden text is commonly processed in some way in order to increase the difficulty of detecting the secret content.

Watermark

Digital watermarking is the method of embedding data into digital multimedia content. This is used to verify the credibility of the content or to recognize the identity of the digital content's owner. A watermark is a “secret message” that is embedded into a “cover message”. Usually, only the knowledge of a secret key allows us to extract the watermark.

Why we use Watermark?

- Ownership assertion.
- Fingerprinting.
- Copy prevention or control.
- Content protection (visible watermarks).
- Authentication.