CRYPTOGRAPHY AND NETWORK SECURITY (CSE 3035)

GP-2 (2*1HR CLASS + 1*2HR PS)

Textbook: Cryptography and Network security
by Williams Stallings (7th Edition)

What is cryptography?

- Cryptography is associated with the process of converting text massage into unintelligible text and vice-versa.

- It is a method of storing and transmitting data in a particular form so that only those for whom it is intended can read and process it.

- Cryptography is mainly concerned with internet security.

Simple Example

123456	213456	546312
↓	↓	↓
ABCDEF	BACDEF	??????

Simple Example

123456	213456	546312
↓	↓	↓
ABCDEF	BACDEF	EDFCAB

What is Internet Security?

- We protect data.

- The dangers are the computers, the computer networks and the biggest of them all is the internet and its likely pitfalls.

- Data security is vital, especially now when all types **of transactions are being carried over internet** to such a large extent.

Need for Security!

- Due to internet taking over the world by storm, there exists many examples about what could happen if there was **insufficient security** built in applications developed.

- The field of security consists of **measures to deter, prevent, detect and correct security violations** that involve the transmission of information.

Unauthorized but is able to monitor the transmission and capture a copy of the file

Silent Features of the modern nature of attacks

- Automating Attacks: The computers and internet have modernized the attacks making them more destructive, like stealing digital money now has become very simple.

- Privacy concerns: Collecting information about people and later misusing it has become a huge problem today, like breaking through the UIADI database will cause havoc.

- Distance is not an issue: People sitting at a single place can attack banks world wide.

Computer Security

It is defined as 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).

(The above definition is from the NIST Computer Security Handbook [NIST95])

Figure 1.1 The Security Requirements Triad

Key objectives of Security

Confidentiality

- A sending secret message/file to B and even if someone else gets it must not know about the contents of the message.
- If the **contents of the message is known by another user C, then confidentiality** is lost. This is termed as **interception**.
- 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 individual's control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.
- Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.

Key objectives of Security

Integrity

- A sending secret message/file to B but **contents of the message are changed** before reaching B. Say A wants to send 100 to B but the message contents are hacked and changed to 1000. This type of security attacks are termed as modification.
- Modification causes loss of message integrity.
- This term covers two related concepts:
  - **Data integrity**: Assures that information (both stored and in transmitted packets) 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.
- Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.

Key objectives of Security

Availability

- Assures that systems work promptly, and service is not denied to authorized users.
- Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.

- These three concepts form what is often referred to as the CIA triad. The three concepts embody the fundamental security objectives for both data and for information and computing services.

Key objectives of Security

Figure 1.1 Essential Network and Computer Security Requirements

Authentication

- A sending secret message/file to B, but how to know that **A is the authentic person**.
- Someone like C can **impersonate A** and send a message to B, this type of security attack is termed as **fabrication**.
- Fabrication happens in the absence of proper authentication mechanisms.
- This is the property of being genuine and being able to be verified and trusted.

Key objectives of Security

Figure 1.1 Essential Network and Computer Security Requirements

Accountability

- This security goal implies that the actions of an entity can be **traced uniquely to that entity**.
- A sending secret message/file to B, **this action can afterwards be traced uniquely.**
- This supports nonrepudiation, 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.

Levels of impact

Three levels of impact on organizations or individuals should there be a breach of security (i.e., a loss of confidentiality, integrity, or availability):

Low, Moderate, and High

CONFIDENTIALITY

- **High:** Student grade information is an asset whose confidentiality is highly important by students. In the United States, the release of such information is regulated by the Family Educational Rights and Privacy Act (FERPA). Grade information should only be available to students, their parents, and employees that require the information to do their job.
- **Moderate:** Student enrollment information may have a moderate confidentiality rating. While still covered by FERPA, this information is seen by more people on a daily basis, is less likely to be targeted than grade information, and results in less damage if disclosed.
- **Low:** Directory information, faculty or departmental lists, may be assigned a low confidentiality rating or indeed no rating. This information is typically freely available to the public and published on a school's Website.

Levels of impact

Three levels of impact on organizations or individuals should there be a breach of security (i.e., a loss of confidentiality, integrity, or availability): **Low, Moderate, and High**

INTEGRITY

- **High:** Several aspects of integrity are illustrated by the example of a hospital patient's allergy information stored in a database. The doctor should be able to trust that the information is correct and updated. Now suppose that an employee (e.g., a nurse) who is authorized to view and update this information deliberately falsifies the data to cause harm to the hospital. The database needs to be restored to a trusted basis quickly, and it should be possible to trace the error back to the person responsible. Patient allergy information is an example of an asset with a high requirement for integrity. Inaccurate information could result in serious harm or death to a patient and expose the hospital to massive liability.
- **Moderate:** An example of an asset that may be assigned a moderate level of integrity requirement is a Website that offers a forum to registered users to discuss some specific topic. Either a registered user or a hacker could falsify some entries or deface the Website. If the forum exists only for the enjoyment of the users, brings in little or no advertising revenue, and is not used for something important such as research, then potential damage is not severe. The Web master may experience some data, financial, and time loss.
- **Low:** An example of a low integrity requirement is an anonymous online poll. Many Web sites, such as news organizations, offer these polls to their users with very few safeguards. However, the inaccuracy and unscientific nature of such polls is well understood.

Levels of impact

Three levels of impact on organizations or individuals should there be a breach of security (i.e., a loss of confidentiality, integrity, or availability): **Low, Moderate, and High**

AVAILABILITY

- **High:** The more critical a component or service, the higher is the level of availability required. Consider a system that provides authentication services for critical systems, applications, and devices. An interruption of service results in the inability for customers to access computing resources and staff to access the resources they need to perform critical tasks. The loss of the service translates into a large financial loss in lost employee productivity and potential customer loss.
- **Moderate:** An example of an asset that would typically be rated as having a moderate availability requirement is a public Website for a university; the Website provides information for current and prospective students and donors. Such a site is not a critical component of the university's information system, but its unavailability will cause some embarrassment.
- **Low:** An online telephone directory lookup application would be classified as a low availability requirement. Although the temporary loss of the application may be an annoyance, there are other ways to access the information, such as a hardcopy directory or the operator.

Computer Security Models

- No Security: Approach to implement no security at all.
- Security through obscurity: A system is secure simply because nobody knows about its existence or contents.
- Host security: The security for each host system is ensured individually.
- Network security: The focus here is to control the network access to different hosts and their services connected through networks.

The OSI security Architecture

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

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

- A systematic way of defining the security requirements of an organization and the approaches taken to satisfy them.
- ITU-T recommendation X-800 , security architecture for OSI defines the systematic approach for organizing the task of providing security.
- It mainly focusses on :

Security Attack

Security Mechanism

Security service

The OSI security Architecture

Security Attack: Action that compromises the security.

Security Mechanism: Detect, prevent, or recover from a security attack.

Security Service: Enhances the security, counter security attacks, and provide the service.

Security Attacks (Technical View)

Passive Attacks

(A passive attack attempts to learn or make use of information from the system but does not affect system resources. They don't involve any modifications.)

Active Attacks

Here the contents of the original massage is modified in some way. An active attack attempts to alter system resources or affect their operation.

Passive Attacks (Interception)

- Again of two types: **release of message contents** and **traffic analysis**.

- A sends a message to B, C may intercept and read the message but cannot modify its contents.

- A sends a message to B but in an encrypted format, here C won't be able to read the message directly but can analyse the various messages coming through the channel and come up with a likely pattern to read the message.

Passive Attacks :
Release of message contents

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.

(a) Release of message contents

Passive Attacks : 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.

(b) Traffic analysis

Passive attacks are very 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

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories:

Masquerade:

A sends a message to B and C trying to pose as A may also send a message, these are called the **masquerade attacks**.

A masquerade takes place when one entity pretends to be a different entity.

(a) Masquerade

Active Attacks

Replay

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
A sends a fund transfer message to B, C copies the message and again resends it to B.

(b) Replay

Active Attacks

Modification of messages

- Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
- A sends a fund transfer message to B and C intercepts and changes the fund amount.

(c) Modification of messages

Active Attacks

Denial of service

- The **denial of service** prevents or inhibits 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 (e.g., the security audit service).
- 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.

(d) Denial of service

Passive Attacks

- No Modification
- Hard to detect
- Neither sender nor receiver is aware
- Prevention is required
- Encryption & Decryption can prevent this

Active Attacks

- Modification happened
- Hard to Prevent
- Detection is possible
- Recovery from any disruption or delays

How attacks are happening?

- They occur due to certain programs devised for generating the security attacks.
- These are termed as virus and worms.
- Virus is a piece of program launched for an application or network level attack which attaches itself to a legitimate program and runs along with it, in this process it self propagates and infects other applications and other users on the same network.
- Worms: It is similar in concept to a virus but different in implementation. A virus modifies a program whereas a worm replicates itself again and again without hampering the program, but in turn utilizes the system resources making the system slow.

- **A MODEL FOR NETWORK SECURITY**

A MODEL FOR NETWORK SECURITY

- A message is to be transferred from one party to another across some sort of Internet service. The two parties, who are the *principals* in this transaction, must cooperate for the exchange to take place.
- A logical information channel is established by defining a route through the Internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.
- Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on.

All the techniques for providing security have two components:

- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.
- A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

A MODEL FOR NETWORK SECURITY

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

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

Network Access Security Model

Network Access Security Model

Network Access Security Model

- A general model of few other situations, which reflects a concern for protecting an information system from unwanted access.
- Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

Network Access Security Model

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

- **Information access threats:** Intercept or modify data on behalf of users who should not have access to that data.
- **Service threats:** Exploit service flaws in computers to inhibit use by legitimate users.

- ❖ Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software.
- ❖ They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.
- ❖ The security mechanisms needed to cope with unwanted access fall into two broad categories.
- ❖ The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access.
- ❖ The second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

Classical Encryption Techniques

1. What Is Cryptography?
2. Applications of Symmetric Key Cryptography
3. What Is Symmetric Key Cryptography?
4. Private-Key Cryptography
5. Types of encryption
6. Advantages of Symmetric Key Cryptography

Cryptography

Types of Encryption

Symmetric Key Cryptography

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.

Symmetric Key Cryptography

There are two requirements for secure use of conventional encryption:

- 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. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

- 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.

Symmetric Key Cryptography

Cryptanalysis and Brute-Force Attack

The objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single ciphertext. 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.

Symmetric Key Cryptography

Private key Cryptography

- Same Key for encryption and decryption means a single point of failure
- Key needs to be always kept secret
- Receiver/Third party can also generate messages with the same key, so authentication issue will arise should the secret key is leaked

Advantages of Symmetric key Cryptography

Faster than Asymmetric

Better Performance Metrics

Optimized for bulk amounts of data

Easier to set-up and implement

Applications of Symmetric Key Cryptography

Banking applications to authenticate ID and transactions

Server/Data Center information can be encrypted at rest

HTTPS encryption with secure all-around browsing

Classical Encryption Techniques

1. What Is Cryptography?
2. Applications of Symmetric Key Cryptography
3. What Is Symmetric Key Cryptography?
4. Private-Key Cryptography
5. Types of encryption
6. Advantages of Symmetric Key Cryptography

Cryptography

Types of Encryption

Symmetric Key Cryptography

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.

Symmetric Key Cryptography

There are two requirements for secure use of conventional encryption:

- 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. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.

- 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.

Symmetric Key Cryptography

Private key Cryptography

- Same Key for encryption and decryption means a single point of failure
- Key needs to be always kept secret
- Receiver/Third party can also generate messages with the same key, so authentication issue will arise should the secret key is leaked

Advantages of Symmetric key Cryptography

Faster than Asymmetric

Better Performance Metrics

Optimized for bulk amounts of data

Easier to set-up and implement

Applications of Symmetric Key Cryptography

Banking applications to authenticate ID and transactions

Server/Data Center information can be encrypted at rest

HTTPS encryption with secure all-around browsing

Symmetric Key Cryptography

Cryptanalysis and Brute-Force Attack (Attacker is known as Cryptanalyst)

The objective of attacking an encryption system is to recover the key in use rather than simply to recover the plaintext of a single ciphertext. 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.

Symmetric Key Cryptography

Cryptographic system are characterized by:

- Type of encryption operations used: **Substitution / Transposition**
- Way in which plaintext is processed: **block / stream**

Classical Substitution Ciphers

- where letters of plaintext are replaced by other letters or by numbers or symbols
- or if plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

Caesar Cipher

- ★ Letters are replaced by other letters or symbols.
- ★ The earlier known and simplest method used be Julius Caesar.
- ★ Replacing each letter of the alphabet with the letter standing three places further down the alphabet.

Example:

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M

13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Caesar Cipher

Algorithm:

For each plaintext letter 'p', substitute the ciphertext letter 'C':

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

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

Encryption & Decryption Example

Plain Text: Cryptography

- Use Caesar enciphering to convert to cipher text.

Cipher Text:

- Use Caesar deciphering to convert to plain text.

- **Key=3 : Caesar Cipher**
- **Key other than 3 (1,2,4,5…): Shift cipher**
- **Key can’t be 0**

Caesar Cipher – Pros and Cons

Pros

1. Simple
2. Easy to implement.

Cons

1. The encryption and decryption algorithms are known.
2. There are only 25 keys to try. (Vulnerable to Brute-force attack)
3. The language of the plaintext is known and easily recognizable.

Cryptanalysis not Decryption

Brute force attack

Ciphertext: SQDYMZK

Shifts	Back	Result
0	[26]	SQDYMZK
1	[25]	TREZNAL
2	[24]	USFAOBM
3	[23]	VTGBPCN
4	[22]	WUHCQDO
5	[21]	XVIDREP
6	[20]	YWJESFQ
7	[19]	ZXKFTGR
8	[18]	AYLGUHS
9	[17]	BZMHVIT
10	[16]	CANIWJU
11	[15]	DBOJXKV
12	[14]	ECPKYLW
13	[13]	FDQLZMX

Shifts	Back	Result
13	[13]	FDQLZMX
14	[12]	GERMANY
15	[11]	HFSNBOZ
16	[10]	IGTOCPA
17	[9]	JHUPDQB
18	[8]	KIVQERC
19	[7]	LJWRFSD
20	[6]	MKXSGTE
21	[5]	NLYTHUF
22	[4]	OMZUIVG
23	[3]	PNAVJWH
24	[2]	QOBWKXI
25	[1]	RPCXLYJ

Monoalphabetic Cipher

- ★ The "cipher" line can be any permutation of the 26 alphabetic characters.

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 S = { a , b , c }
- ★ There are six permutations of S.

Monoalphabetic Cipher

- ★ The "cipher" line can be any permutation of the 26 alphabetic characters.
- ★ This would seem to eliminate brute-force techniques for cryptanalysis.
- ★ A single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.
- ★ English language - Nature of plain text is known.

Monoalphabetic Cipher

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z

Example: Plain Text - Neso

Cipher Text - DULA

Relative frequency of English letters

Monoalphabetic Cipher – Example

GZGEWVGRNCP

CT	G	Z	G	E	W	V	G	R	N	C	P
PT	E		E				E				
PT	E		E			T	E				
PT	E		E			T	E			A	
PT	E		E			T	E		L	A	N
PT	E		E			T	E	P	L	A	N
PT	E	X	E	C	U	T	E	P	L	A	N

Monoalphabetic Cipher – Example

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

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

Monoalphabetic Cipher – Example

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ
t a e e te a that e e a a
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX
e t ta t ha e ee a e th t a
EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ
e e e tat e the t

it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in moscow

Monoalphabetic Cipher – Pros and Cons

Pros

1. Better security than Caesar cipher.

Cons

1. Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.
2. Prone to guessing attack using the English letter frequency of occurrence of letters.
3. A countermeasure is to provide multiple substitutes, known as homophones, for a single letter.

Encrypt the plaintext "Attack postponed to tomorrow and do not use our secret paper until further info" using Monoalphabetic cipher technique.

Secret Key: The quick brown fox jumps over the lazy dog.

Note: Ignore the second and latter occurrence of alphabets in the key.

Playfair Cipher

- ★ Aka Playfair square or Wheatstone-Playfair cipher.
- ★ Manual symmetric encryption technique.
- ★ The first literal digram substitution cipher.
- ★ Invented in 1854 by Charles Wheatstone.
- ★ Bore the name of Lord Playfair for promoting its use.

Playfair Cipher

- ★ Multiple letter encryption cipher.
- ★ Digrams.
- ★ 5 x 5 matrix constructed using a keyword (Ex: Monarchy)

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

Example

Plaintext: attack

Digrams: at ta ck

Plaintext: academy

Digrams: ac ad em yx

Plaintext: balloon

Digrams: ba ll oo n

Digrams: ba lx lo on

Rules for encryption using Playfair Cipher

1. Digrams.
2. Repeating Letters - Filler letter.
3. Same Column | ↓ | Wrap around.
4. Same row | → | Wrap around.
5. Rectangle | ⇆ | Swap

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

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

Digrams

oa
ab
gk
dc
rz
zr
mz
rm

Playfair cipher

- ✓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.
- ✓Furthermore, the relative frequencies of individual letters exhibit a much greater range, making frequency analysis much more difficult.
- ✓It was used as the standard field system by the British Army in World War I, U.S. Army and other Allied forces during World War II.
- ✓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

- ★ Multi-letter cipher.
- ★ Developed by Lester Hill in 1929.
- ★ Encrypts a group of letters : digraph, trigraph or polygraph.

- ★ Review few terminologies from linear algebra.
- ★ Concepts to be known:
  - ○ Matrix arithmetic modulo 26.
  - ○ Square matrix.
  - ○ Determinant.
  - ○ Multiplicative inverse.

The Hill Algorithm

This can be expressed as

$C = E(K,P) = P \times K \bmod 26$

$P = D(K,C) = C\ K^{-1} \bmod 26$

Hill Cipher Example

Question: Encrypt "pay more money" using Hill cipher with key

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

$$(C_1\ C_2\ C_3) = (P_1\ P_2\ P_3) \begin{pmatrix} K_{11} & K_{12} & K_{13} \\ K_{21} & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \bmod 26$$

Encryption

$$C_1 = (P_1K_{11} + P_2K_{21} + P_3K_{31}) \bmod 26$$
$$C_2 = (P_1K_{12} + P_2K_{22} + P_3K_{32}) \bmod 26$$
$$C_3 = (P_1K_{13} + P_2K_{23} + P_3K_{33}) \bmod 26$$

Block ciphers and Data Encryption Standard

Traditional Block Cipher Structure

- Stream Ciphers and Block Ciphers
- Motivation for the Feistel Cipher Structure
- The Feistel Cipher

The Data Encryption Standard

- DES Encryption
- DES Decryption

A DES Example

The Strength of DES

Block ciphers and Data Encryption Standard

BLOCK CIPHER

&

STREAM CIPHER

- Stream Cipher
- Block Cipher
- Difference between Stream & Block Cipher

Stream Cipher

In stream cipher keys and algorithms are applied to each binary digit in a data stream, one bit at a time, rather than encrypting block of data.

One time pad cipher is the best example of stream cipher. Cipher feedback mode is convert block cipher into stream cipher.

Stream Cipher

- A stream cipher operates on plaintext accepting only one bit at a time.
- Stream cipher in which each bit of plain text message XOR with each bit of key to obtain cipher text message.

Stream Cipher

- There is a key stream generator which outputs a stream of bits: k1, k2, k3,...ki and XORed with a stream of plaintext bits p1, p2,...,pi to produce the stream of cipher text bits.

$$Ci = Pi \oplus Ki$$

- During decryption, the cipher text bits are XORed with a same key stream to recover the plain text bits.

$$Pi = Ci \oplus Ki$$

- If the key stream output is random that, it will take harder for a cryptanalyst to break it.

Stream Cipher

Message - sa

Message – Hexa 7361

01110011 8bits

XOR ⊕

01101100

key

0	1	1	1	0	0	1	1
0	1	1	0	1	1	0	0
0	0	0	1	1	1	1	1

C1

Plain

0110 0001 8bits

XOR ⊕

key 01101100

0	1	1	0	0	0	0	1
0	1	1	0	1	1	0	0
0	0	0	0	1	1	0	1

C2

Stream Cipher

Result

0	0	0	1	1	1	1	1

XOR ⊕

Key

01101100

0	0	0	1	1	1	1	1
0	1	1	0	1	1	0	0
0	1	1	1	0	0	1	1

Original message

Result

0	0	0	0	1	1	0	1

C2

XOR ⊕

01101100 key

0	0	0	0	1	1	0	1
0	1	1	0	1	1	0	0
0	1	1	0	0	0	0	1

7361 hexadecimal

Message - sa Original plaint text

Block Cipher

In Block cipher, plain text message devide into fixed size blocks and encrypt each block with some fixed size of key.

Divide each plain text message into block of 64, 128, 256 bits and apply key size 40, 56, 64, 80, 128, 168, 192 and 256 bits which generates cipher text block same as size of plain text block.

Data Encryption Standard (DES) is the best example of block cipher in which each block of 64-bit get encrypted using 56-key bit and cipher text of 64-bit get generated.

Block Cipher

- A receiver side decrypt message with same key to generate plain text.
- Block cipher also uses the concept of key generator.
- Block cipher are used chaining mode, this is because for repeating text pattern, the same cipher block will be generated which can give clue to cryptanalyst regarding what is the original plain text.
- As a chaining mode, previous block is mixed with current block to avoid repeats in patterns. It is more secure.

What is Block Cipher ?

Satishcj isguilty

736174697368636a

0111 0011 0110 0001 0111 0100 0110 1001
0111 0011 0110 1000 0110 0011 0110 1010

Cypher Text

1101 1101 1101 1101 1101 1101 1101 1101
1010 1010 1010 1010 1010 1010 1010 1010

69736775696c7479

0110 1001 0111 0011 0110 0111 0111 0101
0110 1001 0110 1100 0111 0100 0111 1001

Cypher Text

1100 1100 1100 1010 1010 1010 1101 1101
1001 1001 1001 1000 1000 1000 0111 0111

Block Cipher Padding

Satishcj guilty

736174697368636a

64 bits

0111 0011 0110 0001 0111 0100 0110 1001
0111 0011 0110 1000 0110 0011 0110 1010

6775696c7479 6775696c7479**0000**

0110 0111 0111 0101 0110 1001 0110 1100
0111 0100 0111 1001

padding

0110 0111 0111 0101 0110 1001 0110 1100
0111 0100 0111 1001 **0000 0000 0000 0000**

64 bits

1101 1101 1101 1101 1101 1101 1101 1101
1010 1010 1010 1010 1010 1010 1010 1010

1100 1100 1100 1010 1010 1010 1101 1101
1001 1001 1001 1000 1000 1000 0111 0111

Difference between block and stream cipher

No.	Stream Cipher	Block Cipher
1	Key and algorithm applied on each binary digit.	Key and algorithm applied on block of data.
2	Less time consuming in compare to block cipher.	More time consuming in compare to stream cipher.
3	Only one bit encrypting at a time, it is faster than block cipher.	Block of data is encrypting at a time block cipher is slower than stream cipher.
4	It doesn't used chaining mode.	It uses chaining mode.
5	Hardware implementation is easy using stream cipher.	Software implementation is easy using block cipher.
6	One time pad is the best example of stream cipher.	DES is the best example of block cipher.
7	Less Secure in compare to block cipher.	More secure in compare to stream cipher.

Block Cipher

- Most symmetric block ciphers are based on a **Feistel Cipher Structure.**
- Block ciphers look like an extremely large substitution
- Need table of 2^n entries for a n-bit block
- Size of the plaintext and ciphertext must be same
- It should not be used for small size message
- There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular.

Reversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	00
11	01

Irreversible Mapping	
Plaintext	**Ciphertext**
00	11
01	10
10	01
11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is $2^n!$.

Ideal Block Cipher

Figure illustrates the logic of a general substitution cipher for $n = 4$. A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits. The encryption and decryption mappings can be defined by a tabulation. This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. Feistel refers to this as the *ideal block cipher*, because it allows for the maximum number of possible encryption mappings from the plaintext block.

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

Ideal Block Cipher (Issues)

- If a small block size, such as $n = 4$, is used, then the system is equivalent to a classical substitution cipher. Such systems, as we have seen, **are vulnerable to a statistical analysis of the plaintext**. This weakness is not inherent in the use of a substitution cipher but rather results from the **use of a small block size**.
- If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.
- An arbitrary reversible substitution cipher (the ideal block cipher) for a large block size is not practical, however, from an implementation and performance point of view. For such a transformation, the **mapping itself constitutes the key**.
- Consider one particular reversible mapping from plaintext to ciphertext for n = 4. The mapping can be defined by the entries in the second column, which show the value of the ciphertext for each plaintext block. This, in essence, is the key that determines the specific mapping from among all possible mappings. In this case, using this straightforward method of defining the key, the required key length is (4 bits) * (16 rows) = 64 bits. In general, **for an n-bit ideal block cipher, the length of the key defined in this fashion is $n * 2^n$ bits. For a 64-bit block, which is a desirable length to thwart statistical attacks, the required key length is $64 * 2^{64} = 2^{70} \approx 10^{21}$ bits.**

Feistel Structure for Block Ciphers

- Feistel proposed applying two or more simple ciphers in sequence so final result cryptographically stronger than component ciphers
- **n-bit** block length; **k-bit** key length; $\mathbf{2^k}$ transformations (rather than 2n !)
- Feistel cipher alternates: substitutions, transpositions (permutations)
- Approach:
  - Plaintext split into halves
  - Subkeys (or round keys) generated from master key
  - Round function, F, applied to right half
  - Apply substitution on left half using XOR
  - Apply permutation: interchange to halves
- implements Shannon's S-P net concept S-P nets are based on the two primitive cryptographic operations seen before: substitution (S-box) & permutation (P-box)

❑ Feistel Cipher Structure

- Feistel proposed a scheme to produced a block cipher using ***permutation*** and ***substitution*** alternatively.

- The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K_i. The plaintext block is divided into two halves, LE_0 and RE_0.

- The two halves of the data pass through **n** rounds of processing and then combine to produce the ciphertext block.

❑ Working of Feistel Cipher 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***.

Input (plaintext)
LE_0 RE_0
Round 1
F
K_1
LE_1 RE_1
Round 2
F
K_2
LE_2 RE_2
LE_{14} RE_{14}
Round 15
F
K_{15}
LE_{15} RE_{15}
Round 16
F
K_{16}
LE_{16} RE_{16}
LE_{17} RE_{17}
Output (ciphertext)
Output (plaintext)
$RD_{17} = LE_0$ $LD_{17} = RE_0$
$LD_{16} = RE_0$ $RD_{16} = LE_0$
Round 16
F
K_1
$LD_{15} = RE_1$ $RD_{15} = LE_1$
Round 15
F
K_2
$LD_{14} = RE_2$ $RD_{14} = LE_2$
$LD_2 = RE_{14}$ $RD_2 = LE_{14}$
Round 2
F
K_{15}
$LD_1 = RE_{15}$ $RD_1 = LE_{15}$
Round 1
F
K_{16}
$LD_0 = RE_{16}$ $RD_0 = LE_{16}$
Input (ciphertext)

Data Encryption Standard Algorithm(DES)

>> The Data Encryption Standard (DES) was developed in the 1970s by the National Bureau of Standards (NBS)with the help of the National Security Agency (NSA).

>> DES is a block cipher.

>> It encrypts data in block size of 64 bits each.

>> It produces 64 bit of cipher text

>> Same algorithm and key are used for encryption & decryption.

>> The key length is 56 bits. The key originally consists of 64 bits; however, only 56 of these are actually used by the algorithm. Eight bits are used solely for checking parity, and are thereafter discarded. Hence the effective key length is 56 bits.

>> Consists of 16 steps, each of which is called as a round. Each round performs the steps of substitution and transposition

>> Based on two fundamental attributes:

1. Substitution - also called as confusion
2. Transposition - also called as diffusion

Data Encryption Standard Algorithm(DES)

Step 1. Initial Permutation (IP)

Permutation -> transposition

>> It happens only once, before the first round.

>> This is nothing but jugglery of bit positions of the original plain text block.

>> It suggests how the transposition in IP should proceed. For e.g. The IP replaces the first bit of the original plain text block with the 58th bit of the original plain text block, the second with the 50th bit of the original plain text block and so on.

>> After IP is done, the resulting 64-bit permuted block is divided into two 32-bits blocks. (RPT & LPT)

>> 16 rounds are performed on these two blocks.

Data Encryption Standard Algorithm(DES)

16 - Rounds (1 Round Working)

Key -> 56 bit

Step 2.1 Key Transformation

>> For each round, a 56-bit is available.
>> From this 56-bit key, a different 48-bit sub-key is generated during each round using a process called key transformation.
>> This 56-bit key is divided into two halves, each of 28-bits.
>> These halves are circularly shifted left by one or two positions depending on the round.
>> After an appropriate shift, 48 of the 56 bits are selected.
>> The key transformation process involves permutation as well as selection of a 48-bit sub-set of the original 56-bit key, it is called as Compression Permutation.

Data Encryption Standard Algorithm(DES)

16 - Rounds (1 Round Working)

Step 2.2 Expansion Permutation

This happens as follows:

>> The 32-bit RPT is divided into 8 blocks, each consisting of 4 bits.

>> Next, each 4-bit block is then expanded to a corresponding 6-bit block. i.e. per 4-bit block, 2 more bits added. They are actually, the repeated first and the fourth bits of the 4-bit block.

$8 \times 6 = 48$ bit RPT

Data Encryption Standard Algorithm(DES)

16 - Rounds (1 Round Working)

Step 2.2 Expansion Permutation

>> Firstly, the key transformation process compresses the 56-bit key to 48 bits.
>> Then the expansion permutation process expands the 32 bit RPT to 48 bits.
>> Now the 48-bit key is XORed with 48-bit RPT.
>> And the resulting output is given to the next step i.e. S-box Substitution.

Data Encryption Standard Algorithm(DES)

16 - Rounds (1 Round Working)

Step 2.3 S-Box Substitution

>> It is a process that accepts the 48-bit input from the XOR operation involving the compressed key and expanded RPT, and produces a 32 bit output using the substitution technique.

>> The substitution is performed by 8 substitution boxes called as S-Boxes.

>> Each of the eight S-boxes has a 6-bit input and a 4-bit output.

>> The 48-bit input block is divided into 8 sub-blocks and each such block is given to a S-box.

>> The output of all the S-Boxes are then combined to form a 32-bit block, which is given to the next stage of a round, the P-box permutation.

Data Encryption Standard Algorithm(DES)

16 - Rounds (1 Round Working)

Step 2.4 P-Box Permutation

>> The output of S-box consists of 32 bits, these are permuted using a P-box.

>> This involves simple permutation (i.e. swapping of each bit with another bit, without any expansion or compression)

>> This is called as P-Box Permutation.

Data Encryption Standard Algorithm(DES)

16 - Rounds (1 Round Working)

Step 2.5 X-OR & Swap

>> All these operations are performed only on the 32-bit right half portion of the 64-bit original plain text (i.e. on the RPT).
>> The left half portion (i.e. LPT) was untouched.
>> The LPT is now XORed with the output produced by P-box permutation.
>> The result of this XOR operation becomes the new right half (RPT).
>> The old right half (i.e. RPT) becomes the new left half (i.e. LPT), in a process of swapping.

Data Encryption Standard Algorithm(DES)

Step 3. Final Permutation (FP)

>> At the end of the 16 rounds, the Final Permutation is performed (only Once).
>> This is a simple transposition.
>> For ex. The 40th input bit takes the position of the 1st output bit and so on.
>> The output of final permutation is the 64-bit encrypted block.

Data Encryption Standard Algorithm(DES)

DES Decryption

>> The same algorithm used for encryption also works for decryption i.e. the algorithm is reversible.
>> The only difference between the encryption and the decryption process is the reversal of key portions.
>> If the original key K was divided into K1,K2,K3,....,K16 for the 16 encryption rounds, then for decryption, the key should be used as K16, K15,K14,.....,K1.

The strength of DES

>> The inner workings of the DES algorithm are known to the general public. Therefore the strength of DES lies in its Key, which must be secret.
>> DES uses 56-bit keys. (meaning that there are 72,057,594,037,927,936 possible keys.)
>> Thus, it seems that a brute-force attack on DES is impractical.

Variations of DES

Two major variations of DES

1. Double DES
2. Triple DES
   - 2.1 Triple DES that uses three keys
   - 2.2 Triple DES that uses two keys

Origin of DES

General Depiction of DES Encryption Algorithm

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

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}

Then the permutation $X = (\mathrm{IP}(M)$ 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

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^{-1})

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

Single Round of DES Algorithm

DES Key Schedule Calculation

(a) Input Key

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	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

(b) Permuted Choice One (PC-1)

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

(c) Permuted Choice Two (PC-2)

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

(d) Schedule of Left Shifts

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

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

- 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 S_i.
- 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.

Definition of DES S-Boxes

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

- 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 S_i.
- 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.

(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

Avalanche Effect

- Aim: small change in key (or plaintext) produces large change in ciphertext
- Avalanche effect is present in DES (good for security)
- Following examples show the number of bits that change in output when two different inputs are used, differing by 1 bit

Plaintext 1: 02468aceeca86420
Plaintext 2: 12468aceeca86420
Ciphertext difference: 32 bits

If key difference by 1-bit

Key 1: 0f1571c947d9e859
Key 2: 1f1571c947d9e859
Ciphertext difference: 30

STRENGTH OF DES: KEY SIZE & NATURE OF THE ALGORITHM

The Use of 56-Bit Keys

- With a key length of 56 bits, there are possible 2^{56} keys, which is approximately $7.2*10^{16}$ keys. Thus, 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.
- DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose "DES cracker" machine that was built for less than $250,000. The attack took less than three days. The EFF has published a detailed description of the machine, enabling others to build their own cracker [EFF98].And, of course, hardware prices will continue to drop as speeds increase, making DES virtually worthless.
- Fortunately, there are a number of alternatives to DES, the most important of which are AES and triple DES.

The Nature of the DES Algorithm

- Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration. Because the design criteria for these boxes, and indeed for the entire algorithm, were not made public, there is a suspicion that the boxes were constructed in such a way that cryptanalysis is possible for an opponent who knows the weaknesses in the S-boxes.
- This assertion is tantalizing, and over the years a number of regularities and unexpected behaviors of the S-boxes have been discovered. Despite this, no one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.

Two most powerful and promising cryptanalysis approaches for DES:

- Differential cryptanalysis
- Linear cryptanalysis

Differential Cryptanalysis

- Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 2^{55} (2^{47}) encryptions.
- The differential cryptanalysis attack is complex and provides a complete description.
- The rationale behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each
- round of the cipher, instead of observing the evolution of a single text block.
- Consider the original plaintext block to consist of two halves m_0 and m_1. Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the subkey for this round. So, at each round, only one new 32-bit block is created.

If we label each new block m_i, then the intermediate message halves are related as follows:

$$m_{i+1} = m_{i-1} \oplus \mathrm{f}(m_i, K_i), \qquad i = 1, 2, \ldots, 16$$

In differential cryptanalysis, we start with two messages, m and m', with a known XOR difference $\Delta m = m \oplus m'$, and consider the difference between the intermediate message halves: $\Delta m_i = m_i \oplus m'_i$. Then we have

$$\begin{aligned}\Delta m_{i+1} &= m_{i+1} \oplus m'_{i+1} \\ &= [m_{i-1} \oplus \mathrm{f}(m_i, K_i)] \oplus [m'_{i-1} \oplus \mathrm{f}(m'_i, K_i)] \\ &= \Delta m_{i-1} \oplus [\mathrm{f}(m_i, K_i) \oplus \mathrm{f}(m'_i, K_i)]\end{aligned}$$

Linear Cryptanalysis

- A more recent development is linear cryptanalysis.
- This attack is based on finding linear approximations to describe the transformations performed in DES.
- Linear cryptanalysis is capable of breaking DES in less than 2^{43} encryptions.

We now give a brief summary of the principle on which linear cryptanalysis is based. For a cipher with n-bit plaintext and ciphertext blocks and an m-bit key, let the plaintext block be labeled $P[1], \ldots P[n]$, the cipher text block $C[1], \ldots C[n]$, and the key $K[1], \ldots, K[m]$. Then define

$$A[i, j, \ldots, k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

The objective of linear cryptanalysis is to find an effective *linear* equation of the form:

$$P[\alpha_1, \alpha_2, \ldots, \alpha_a] \oplus C[\beta_1, \beta_2, \ldots, \beta_b] = K[\gamma_1, \gamma_2, \ldots, \gamma_c]$$

(where $x = 0$ or 1; $1 \leq a; b \leq n; c \leq m$; and where the α, β, and γ terms represent fixed, unique bit locations) that holds with probability $p \neq 0.5$. The further p is from 0.5, the more effective the equation. Once a proposed relation is determined, the procedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext–ciphertext pairs. If the result is 0 more than half the time, assume $K[\gamma_1, \gamma_2, \ldots, \gamma_c] = 0$. If it is 1 most of the time, assume $K[\gamma_1, \gamma_2, \ldots, \gamma_c] = 1$. This gives us a linear equation on the key bits. Try to get more such relations so that we can solve for the key bits. Because we are dealing with linear equations, the problem can be approached one round of the cipher at a time, with the results combined.

Some Preliminary Examples of DES

DES works on bits, or binary numbers--the 0s and 1s common to digital computers. Each group of four bits makes up a hexadecimal, or base 16, number. Binary "0001" is equal to the hexadecimal number "1", binary "1000" is equal to the hexadecimal number "8", "1001" is equal to the hexadecimal number "9", "1010" is equal to the hexadecimal number "A", and "1111" is equal to the hexadecimal number "F".

DES works by encrypting groups of 64 message bits, which is the same as 16 hexadecimal numbers. To do the encryption, DES uses "keys" where are also *apparently* 16 hexadecimal numbers long, or *apparently* 64 bits long. However, every 8th key bit is ignored in the DES algorithm, so that the effective key size is 56 bits. But, in any case, 64 bits (16 hexadecimal digits) is the round number upon which DES is organized.

For example, if we take the plaintext message "8787878787878787", and encrypt it with the DES key "0E329232EA6D0D73", we end up with the ciphertext "0000000000000000".

If the ciphertext is decrypted with the same secret DES key "0E329232EA6D0D73", the result is the original plaintext "8787878787878787".

How DES Works in Detail

DES is a block cipher--meaning it operates on plaintext blocks of a given size (64-bits) and returns ciphertext blocks of the same size. Thus DES results in a permutation among the 2^64 (read this as: "2 to the 64th power") possible arrangements of 64 bits, each of which may be either 0 or 1. Each block of 64 bits is divided into two blocks of 32 bits each, a left half block L and a right half R.

Example: Let M be the plain text message

M = 0123456789ABCDEF,

where M is in hexadecimal (base 16) format.

Rewriting M in binary format, we get the 64-bit block of text:

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

L = 0000 0001 0010 0011 0100 0101 0110 0111

R = 1000 1001 1010 1011 1100 1101 1110 1111

The first bit of **M** is "0". The last bit is "1". We read from left to right.

DES operates on the 64-bit blocks using *key* sizes of 56- bits. The keys are actually stored as being 64 bits long, but every 8th bit in the key is not used (i.e. bits numbered 8, 16, 24, 32, 40, 48, 56, and 64). However, we will nevertheless number the bits from 1 to 64, going left to

right, in the following calculations. But, as you will see, the eight bits just mentioned get eliminated when we create sub-keys.

Example: Let **K** be the hexadecimal key

K = 133457799BBCDFF1

This gives us as the binary key (setting 1 = 0001, 3 = 0011, etc., and grouping together every eight bits, of which the last one in each group will be unused):

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

The DES algorithm uses the following steps:

Step 1: Encode each 64-bit block of data

There is an *initial permutation* **IP** of the 64 bits of the message data **M**. This rearranges the bits according to the following table, where the entries in the table show the new arrangement of the bits from their initial order. The 58th bit of **M** becomes the first bit of **IP**. The 50th bit of **M** becomes the second bit of **IP**. The 7th bit of **M** is the last bit of **IP**.

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

Example: Applying the initial permutation to the block of text **M**, given previously, we get

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Here the 58th bit of **M** is "1", which becomes the first bit of **IP**. The 50th bit of **M** is "1", which becomes the second bit of **IP**. The 7th bit of **M** is "0", which becomes the last bit of **IP**.

Next divide the permuted block **IP** into a left half L_0 of 32 bits, and a right half R_0 of 32 bits.

Example: From **IP**, we get L_0 and R_0

L_0 = 1100 1100 0000 0000 1100 1100 1111 1111

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

Step 2: Create 16 sub-keys, each of which is 48-bits long

The 64-bit key is permuted according to the following table, **PC-1**. Since the first entry in the table is "57", this means that the 57th bit of the original key **K** becomes the first bit of the permuted key **K**+. The 49th bit of the original key becomes the second bit of the permuted key. The 4th bit of the original key is the last bit of the permuted key. Note only 56 bits of the original key appear in the permuted key.

PC-1:

Permuted Choice One (PC-1)

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

Example: From the original 64-bit key

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

We get the 56-bit permutation

K+ = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Next, split this key into left and right halves, C_0 and D_0, where each half has 28 bits.

Example: From the permuted key **K**+, we get

C_0 = 1111000 0110011 0010101 0101111

D_0 = 0101010 1011001 1001111 0001111

With C_0 and D_0 defined, we now create sixteen blocks C_n and D_n, $1<=n<=16$. Each pair of blocks C_n and D_n is formed from the previous pair C_{n-1} and D_{n-1}, respectively, for n = 1, 2, ..., 16, using the following schedule of "left shifts" of the previous block. To do a left shift, move each bit one place to the left, except for the first bit, which is cycled to the end of the block.

Schedule of Left Shifts

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

This means, for example, $\boldsymbol{C_3}$ and $\boldsymbol{D_3}$ are obtained from $\boldsymbol{C_2}$ and $\boldsymbol{D_2}$, respectively, by two left shifts, and $\boldsymbol{C_{16}}$ and $\boldsymbol{D_{16}}$ are obtained from $\boldsymbol{C_{15}}$ and $\boldsymbol{D_{15}}$, respectively, by one left shift. In all cases, by a single left shift is meant a rotation of the bits one place to the left, so that after one left shift the bits in the 28 positions are the bits that were previously in positions 2, 3,..., 28, 1.

Example: From original pair pair $\boldsymbol{C_0}$ and $\boldsymbol{D_0}$ we obtain:

$\boldsymbol{C_0}$ = 1111000011001100101010101111

$\boldsymbol{D_0}$ = 0101010101100110011110001111

$\boldsymbol{C_1}$ = 1110000110011001010101011111

$\boldsymbol{D_1}$ = 1010101011001100111100011110

$\boldsymbol{C_2}$ = 1100001100110010101010111111

$\boldsymbol{D_2}$ = 0101010110011001111000111101

$\boldsymbol{C_3}$ = 0000110011001010101011111111

$\boldsymbol{D_3}$ = 0101011001100111100011110101

$\boldsymbol{C_4}$ = 0011001100101010101111111100

$\boldsymbol{D_4}$ = 0101100110011110001111010101

$\boldsymbol{C_5}$ = 1100110010101010111111110000

$\boldsymbol{D_5}$ = 0110011001111000111101010101

$\boldsymbol{C_6}$ = 0011001010101011111111000011

$\boldsymbol{D_6}$ = 1001100111100011110101010101

$\boldsymbol{C_7}$ = 1100101010101111111100001100

$\boldsymbol{D_7}$ = 0110011110001111010101010110

$\boldsymbol{C_8}$ = 0010101010111111110000110011

$\boldsymbol{D_8}$ = 1001111000111101010101011001

$\boldsymbol{C_9}$ = 0101010101111111100001100110

D_9 = 0011110001111010101010110011

C_{10} = 0101010111111110000110011001

D_{10} = 1111000111101010101011001100

C_{11} = 0101011111111000011001100101

D_{11} = 1100011110101010101100110011

C_{12} = 0101111111100001100110010101

D_{12} = 0001111010101010110011001111

C_{13} = 0111111110000110011001010101

D_{13} = 0111101010101011001100111100

C_{14} = 1111111000011001100101010101

D_{14} = 1110101010101100110011110001

C_{15} = 1111100001100110010101010111

D_{15} = 1010101010110011001111000111

C_{16} = 1111000011001100101010101111

D_{16} = 0101010101100110011110001111

We now form the keys K_n, for 1<=n<=16, by applying the following permutation table to each of the concatenated pairs C_nD_n. Each pair has 56 bits, but **PC-2** only uses 48 of these.

Permuted Choice Two (PC-2)

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

Therefore, the first bit of K_n is the 14th bit of C_nD_n, the second bit the 17th, and so on, ending with the 48th bit of K_n being the 32th bit of C_nD_n.

Example: For the first key we have

C_1D_1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

which, after we apply the permutation **PC-2**, becomes

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

For the other keys we have

K_2 = 011110 011010 111011 011001 110110 111100 100111 100101

K_3 = 010101 011111 110010 001010 010000 101100 111110 011001

K_4 = 011100 101010 110111 010110 110110 110011 010100 011101

K_5 = 011111 001110 110000 000111 111010 110101 001110 101000

K_6 = 011000 111010 010100 111110 010100 000111 101100 101111

K_7 = 111011 001000 010010 110111 111101 100001 100010 111100

K_8 = 111101 111000 101000 111010 110000 010011 101111 111011

K_9 = 111000 001101 101111 101011 111011 011110 011110 000001

K_{10} = 101100 011111 001101 000111 101110 100100 011001 001111

K_{11} = 001000 010101 111111 010011 110111 101101 001110 000110

K_{12} = 011101 010111 000111 110101 100101 000110 011111 101001

K_{13} = 100101 111100 010111 010001 111110 101011 101001 000001

K_{14} = 010111 110100 001110 110111 111100 101110 011100 111010

K_{15} = 101111 111001 000110 001101 001111 010011 111100 001010

K_{16} = 110010 110011 110110 001011 000011 100001 011111 110101

Step 3: Iterations (16-Rounds)

We now proceed through 16 iterations, for 1<=n<=16, using a function f which operates on two blocks

- A data block of 32 bits
- A key K_n of 48 bits
- To produce a block of 32 bits.

Then for **n** going from 1 to 16 we calculate

$L_n = R_{n-1}$
$R_n = L_{n-1}$ **XOR** $f(R_{n-1},K_n)$

This results in a final block, for n = 16, of $L_{16}R_{16}$. That is, in each iteration, we take the right 32 bits of the previous result and make them the left 32 bits of the current step. For the right 32 bits in the current step, we XOR the left 32 bits of the previous step with the calculation f.

A. Expansion Permutation:

To calculate f, we first expand each block R_{n-1} from 32 bits to 48 bits. This is done by using a selection table that repeats some of the bits in R_{n-1} . We'll call the use of this selection table the function E. Thus $E(R_{n-1})$ has a 32 bit input block, and a 48 bit output block.

Let E be such that the 48 bits of its output, written as 8 blocks of 6 bits each, are obtained by selecting the bits in its inputs in order according to the following table:

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

Example: We calculate $E(R_0)$ from R_0 as follows:

R_0 = 1111 0000 1010 1010 1111 0000 1010 1010

$E(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

(Note that each block of 4 original bits has been expanded to a block of 6 output bits.)

B. XOR:

Next in the f calculation, we XOR the output $\mathbf{E}(R_{n-1})$ with the key K_n:

$$K_n \text{ XOR } \mathbf{E}(R_{n-1}).$$

Example: For K_1 , $\mathbf{E}(R_0)$, we have

K_1 = 000110 110000 001011 101111 111111 000111 000001 110010

$\mathbf{E}(R_0)$ = 011110 100001 010101 010101 011110 100001 010101 010101

K_1 XOR $\mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

C. S-Box Substitution:

We have not yet finished calculating the function f. To this point we have expanded R from 32 bits to 48 bits, using the selection table, and XORed the result with the key K_n. We now have 48 bits, or eight groups of six bits. We now do something strange with each group of six bits: we use them as addresses in tables called "**S-boxes**". Each group of six bits will give us an address in a different **S**-box. Located at that address will be a 4-bit number. This 4-bit number will replace the original 6-bits. The net result is that the eight groups of 6-bits are transformed into eight groups of 4-bits (the 4-bit outputs from the **S** boxes) for 32 bits total.

Write the previous result, which is 48 bits, in the form:

K_n XOR $\mathbf{E}(R_{n-1}) = B_1B_2B_3B_4B_5B_6B_7B_8$,

where each B_i is a group of six bits. We now calculate

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$

Where, $S_i(B_i)$ referred to the output of the i-th **S** box.

To repeat, each of the functions $S1, S2,..., S8$, takes a 6-bit block as input and yields a 4-bit block as output. The table to determine S_1 is shown and explained below:

Example: If S_1 is the function defined in this table and B is a block of 6 bits, then $S_1(B)$ is determined as follows: The first and last bits of B represent in base 2 a number in the decimal range 0 to 3 (or binary 00 to 11). Let that number be i. The middle 4 bits of B represent in base 2 a number in the decimal range 0 to 15 (binary 0000 to 1111). Let that number be j. Look up in the table the number in the i-th row and j-th column. It is a number in the range 0 to 15 and is uniquely represented by a 4 bit block. That block is the output $S_1(B)$ of S_1 for the input B.

For example, for input block B = 011011 the first bit is "0" and the last bit "1" giving 01 as the row. This is row 1. The middle four bits are "1101". This is the binary equivalent of decimal 13, so the column is column number 13. In row 1, column 13 appears 5. This determines the output; 5 is binary 0101, so that the output is 0101. Hence $S_1(011011) = 0101$.

S1

Column Number

Row No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Definition of DES S-Boxes

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

Example: For the first round, we obtain as the output of the eight **S** boxes:

K_1 XOR $\mathbf{E}(R_0)$ = 011000 010001 011110 111010 100001 100110 010100 100111.

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

D. P-Box Substitution:

The permutation **P** is defined in the following table. **P** yields a 32-bit output from a 32-bit input by permuting the bits of the input block.

(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

Example: From the output of the eight **S** boxes:

$S_1(B_1)S_2(B_2)S_3(B_3)S_4(B_4)S_5(B_5)S_6(B_6)S_7(B_7)S_8(B_8)$ = 0101 1100 1000 0010 1011 0101 1001 0111

we get f = **0010 0011 0100 1010 1010 1001 1011 1011**

E. XOR & SWAP:

$R_1 = L_0$ XOR $f(R_0, K_1)$

1100 1100 0000 0000 1100 1100 1111 1111

XOR

0010 0011 0100 1010 1010 1001 1011 1011

R_1 = 1110 1111 0100 1010 0110 0101 0100 0100

$L_1 = R_0$ = 1111 0000 1010 1010 1111 0000 1010 1010

In the next round, we will have $L_2 = R_1$, which is the block we just calculated, and then we must calculate $R_2 = L_1$ ***XOR*** $f(R_1, K_2)$, and so on for 16 rounds. At the end of the sixteenth round we have the blocks L_{16} and R_{16}. We then ***reverse*** the order of the two blocks into the 64-bit block $R_{16}L_{16}$

Example: If we process all 16 blocks using the method defined previously, we get, on the 16th round,

L_{16} = 0100 0011 0100 0010 0011 0010 0011 0100

R_{16} = 0000 1010 0100 1100 1101 1001 1001 0101

We reverse the order of these two blocks and apply the final permutation to

$R_{16}L_{16}$ = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

And apply a final permutation $\mathbf{IP^{-1}}$ as defined by the following table:

(b) Inverse Initial Permutation (IP^{-1})

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

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

In hexadecimal format: 85E813540F0AB405.

This is the encrypted form of **M=0123456789ABCDEF**

C = 85E813540F0AB405

Decryption is simply the inverse of encryption, following the same steps as above, but reversing the order in which the sub-keys are applied.

ADVANCED ENCRYPTION STANDARD

With DES being an old standard, computers of today can break the security in seconds.

Chronology of DES Cracking	
Broken for the first time	1997
Broken in 56 hours	1998
Broken in 22 hours and 15 minutes	1999
Capable of broken in 5 minutes	2021

- ✓ The AES algorithm (Rijndael Algo) is a symmetric block cipher algorithm that takes a plaintext block size of 128 bits (16 bytes) and converts them to ciphertext using key length of 16, 24, or 32 bytes (128, 192, or 256 bits).

- ✓ The algorithm is referred to as AES-128, AES-192, or AES-256, depending on the key length.

- ✓ The input to the encryption and decryption algorithms is a single 128-bit block.

- ✓ AES uses arithmetic in the finite field $GF(2^8)$ with the irreducible polynomial

$$m(x)=x^8+ x^4+x^3+x+1$$

Features of AES

1. It uses **Substitution and Permutations**, also called SP Networks.
2. A single key is expanded to be used in **multiple rounds**.
3. AES performs on **byte data**, instead of bit data.
4. No. of rounds is dependent on **key length**.

128-bit Key Length	→	10 rounds
192-bit Key Length	→	12 rounds
256-bit Key Length	→	14 rounds

How Does AES Work?

Manner of Storage

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

- Everything is stored in a **4x4 matrix** format.
- Known as **state array**.
- Each round takes state array as input and gives **similar output**.
- **16-byte matrix**, with each cell representing one byte.
- **4 bytes = 1 word**, so each state array has 4 words.

How Does AES Work?

Key Expansion

- The key is expanded to **n+1 rounds**, with n being no. of rounds.
- **4 words** in each key.
- Each key is used for a **single round**.

K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7
K_8	K_9	K_{10}	K_{11}
K_{12}	K_{13}	K_{14}	K_{15}

How Does AES Work?

How Does AES Work?

Adding the Round Key

- Plaintext is stored in a **state array** and XOR'd with K_0.
- Performed only **once** per block.
- Will be performed once again at the **end of each round**.

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

$\oplus$

XOR

K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7
K_8	K_9	K_{10}	K_{11}
K_{12}	K_{13}	K_{14}	K_{15}

How Does AES Work?

Byte Substitution (Sub-Bytes)

- Clever **lookup table** to map one byte to another.
- Works as a **substitution** table.
- Each byte is converted to **hexadecimal**, with 2 equal parts.
- First part is the **row**; second part is **column**.

How Does AES Work?

Shift Rows

- Shifting **row elements** among each other
- Increases **complexity** of the algorithm
- First row is to be skipped, second row moves **1 place**, third row moves **2 places** and last row moves **3 places**

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

→

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

How Does AES Work?

Mix Columns

- Multiply each column with a **constant matrix**
- **Resultant** matrix forms the new column
- **Not to be done** in the last round

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \times \begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} NC_0 \\ NC_1 \\ NC_2 \\ NC_3 \end{bmatrix}$$

Constant Matrix Old Column New Column

How Does AES Work?

Add Round Key

- The **expanded key** is used with the result matrix
- If it's last round, the result is **Ciphertext**
- If not the last round, result is **input for next round**

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

$\oplus$

XOR

K_0	K_1	K_2	K_3
K_4	K_5	K_6	K_7
K_8	K_9	K_{10}	K_{11}
K_{12}	K_{13}	K_{14}	K_{15}

Differences Between AES & DES

DES	AES
• Key Length – **56 bits**	• Key Length – **128/192/256 bits**
• Block Size – **64 bits**	• Block Size – **128 bits**
• Fixed no of rounds - **16**	• No. of rounds dependent on key length
• Comparatively **slower**	• Comparatively **faster**

How Does AES Work? - Example

Plaintext – Two One Nine Two

T	w	o		O	n	e		N	i	n	e		T	w	o
54	77	6F	20	4F	6E	65	20	43	69	6E	25	20	54	77	6F

Plaintext in Hex Format

54 77 6F 20 4F 6E 65 20 43 69 6E 25 20 54 77 6F

Encryption Key – Thats my Kung Fu

T	h	a	t	s		m	y		K	u	n	g		F	u
54	68	61	74	73	20	6D	79	20	4B	75	6E	67	20	46	75

Encryption Key in Hex Format

54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75

Keys generated for every round

- Round 0: 54 68 61 74 73 20 6D 79 20 4B 75 6E 67 20 46 75
- Round 1: E2 32 FC F1 91 12 91 88 B1 59 E4 E6 D6 79 A2 93
- Round 2: 56 08 20 07 C7 1A B1 8F 76 43 55 69 A0 3A F7 FA
- Round 3: D2 60 0D E7 15 7A BC 68 63 39 E9 01 C3 03 1E FB
- Round 4: A1 12 02 C9 B4 68 BE A1 D7 51 57 A0 14 52 49 5B
- Round 5: B1 29 3B 33 05 41 85 92 D2 10 D2 32 C6 42 9B 69
- Round 6: BD 3D C2 B7 B8 7C 47 15 6A 6C 95 27 AC 2E 0E 4E
- Round 7: CC 96 ED 16 74 EA AA 03 1E 86 3F 24 B2 A8 31 6A
- Round 8: 8E 51 EF 21 FA BB 45 22 E4 3D 7A 06 56 95 4B 6C
- Round 9: BF E2 BF 90 45 59 FA B2 A1 64 80 B4 F7 F1 CB D8
- Round 10: 28 FD DE F8 6D A4 24 4A CC C0 A4 FE 3B 31 6F 26

Add Round Key – Round 0

54	4F	4E	20
77	6E	69	54
6F	65	6E	77
20	20	65	6F

Plaintext

XOR

54	73	20	67
68	20	4B	20
61	6D	75	46
74	79	6E	75

Round 0 Key

00	3C	63	47
1F	4E	22	74
0E	08	1B	31
54	59	0B	1A

New State Array

Substitution of Bytes

Substitute each byte with corresponding element in the **Substitution Box**

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

New State Array

Shift Rows

1st row is shifted to the left by 0 places, 2nd row by 1 place, 3rd row by 2 places and last row by 3 places.

63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

Old State Array

→

63	EB	9F	A0
2F	93	92	C0
AF	C7	AB	30
A2	20	CB	2B

New State Array

Mix Columns

Multiply the state array with a constant matrix

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

Constant Matrix

X

63	EB	9F	A0
2F	93	92	C0
AF	C7	AB	30
A2	20	CB	2B

Old State Array

→

BA	84	E8	1B
75	A4	8D	40
F4	8D	06	7D
7A	32	0E	5D

New State Array

Add Round Key – Round 1

BA	84	E8	1B
75	A4	8D	40
F4	8D	06	7D
7A	32	0E	5D

Old State Array

⊕

XOR

E2	91	B1	D6
32	12	59	79
FC	91	E4	A2
F1	88	E6	93

Round 1 Key

58	15	59	CD
47	B6	D4	39
08	1C	E2	DF
8B	BA	E8	CE

New State Array

New State Array

58	15	59	CD
47	B6	D4	39
08	1C	E2	DF
8B	BA	E8	CE

AES Output after 1 Round

58 47 08 8B 15 B6 1C BA 59 D4 E2 E8 CD 39 DF CE

↓

Input for Round 2

Final State Array after Round 10

29	57	40	1A
C3	14	22	02
50	20	99	D7
5F	F6	B3	3A

AES Final Output

29 C3 50 5F 57 14 20 F6 40 22 99 B3 1A 02 D7 3A

↓

Ciphertext

PSEUDO RANDOM NUMBER GENERATOR
&
STREAM CIPHER

A Sample Generator

$$x_n = f(x_{n-1}, x_{n-2}, \ldots)$$

- For example,

$$x_n = 5x_{n-1} + 1 \mod 16$$

- Starting with x_0=5:

$$x_1 = 5(5) + 1 \mod 16 = 26 \mod 16 = 10$$

- The first 32 numbers obtained by the above procedure 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5 10, 3, 0, 1, 6, 15, 12, 13, 2, 11, 8, 9, 14, 7, 4, 5.

- By dividing x's by 16:
  0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125, 0.6250, 0.1875, 0.0000, 0.0625, 0.3750, 0.9375, 0.7500, 0.8125, 0.1250, 0.6875, 0.5000, 0.5625, 0.8750, 0.4375, 0.2500, 0.3125.

Random numbers play an important role in the use of encryption for various network security applications. In this section, we provide a brief overview of the use of random numbers in cryptography and network security and then focus on the principles of pseudorandom number generation.

The Use of Random Numbers

A number of network security algorithms and protocols based on cryptography make use of random binary numbers. For example,

- Key distribution and reciprocal (mutual) authentication schemes, such as those discussed in Chapters 14 and 15. In such schemes, two communicating parties cooperate by exchanging messages to distribute keys and/or authenticate each other. In many cases, nonces are used for handshaking to prevent replay attacks. The use of random numbers for the nonces frustrates an opponent's efforts to determine or guess the nonce, in order to repeat an obsolete transaction.
- Session key generation. We will see a number of protocols in this book where a secret key for symmetric encryption is generated for use for a particular transaction (or session) and is valid for a short period of time. This key is generally called a session key.
- Generation of keys for the RSA public-key encryption algorithm (described in Chapter 9).
- Generation of a bit stream for symmetric stream encryption (described in this chapter).

These applications give rise to two distinct and not necessarily compatible requirements for a sequence of random numbers: randomness and unpredictability.

Randomness Traditionally, the concern in the generation of a sequence of allegedly random numbers has been that the sequence of numbers be random in some well-defined statistical sense. The following two criteria are used to validate that a sequence of numbers is random:

- **Uniform distribution:** The distribution of bits in the sequence should be uniform; that is, the frequency of occurrence of ones and zeros should be approximately equal.
- **Independence:** No one subsequence in the sequence can be inferred from the others.

Unpredictability In applications such as reciprocal authentication, session key generation, and stream ciphers, the requirement is not just that the sequence of numbers be statistically random but that the successive members of the sequence are unpredictable. With "true" random sequences, each number is statistically independent of other numbers in the sequence and therefore unpredictable.

TRNG = true random number generator
PRNG = pseudorandom number generator
PRF = pseudorandom function

Figure 8.1 Random and Pseudorandom Number Generators

TRNG

- A TRNG takes as input a source that is effectively random; the source is often referred to as an **entropy source**.
- In essence, the entropy source is drawn from the physical environment of the computer and could include things such as keystroke timing patterns, disk electrical activity, mouse movements, and instantaneous values of the system clock.
- The sources, or combination of sources, serve as input to an algorithm that produces random binary output.
- The TRNG may simply involve conversion of an analog source to a binary output.

PRNG

- In contrast, a PRNG takes as input a fixed value, called the seed, and produces a sequence of output bits using a deterministic algorithm. Quite often, the seed is generated by a TRNG.
- Typically, as shown, there is some feedback path by which some of the results of the algorithm are fed back as input as additional output bits are produced.
- The important thing to note is that the output bit stream is determined solely by the input value or values, so that an adversary who knows the algorithm and the seed can reproduce the entire bit stream.
- An algorithm that is used to produce an open-ended sequence of bits is referred to as a PRNG. A common application for an open-ended sequence of bits is as input to a symmetric stream cipher.

Generation of Seed Input to PRNG

PRF

- A PRF is used to produce a pseudorandom string of bits of some fixed length. Examples are symmetric encryption keys and nonces.
- Typically, the PRF takes as input a seed plus some context specific values, such as a user ID or an application ID.
- Other than the number of bits produced, there is no difference between a PRNG and a PRF. The same algorithms can be used in both applications. Both require a seed and both must exhibit randomness and unpredictability.

PSEUDORANDOM NUMBER GENERATORS

Linear Congruential Generators

A widely used technique for pseudorandom number generation is an algorithm first proposed by Lehmer [LEHM51], which is known as the linear congruential method. The algorithm is parameterized with four numbers, as follows:

m	the modulus	$m > 0$
a	the multiplier	$0 < a < m$
c	the increment	$0 \leq c < m$
X_0	the starting value, or seed	$0 \leq X_0 < m$

The sequence of random numbers $\{X_n\}$ is obtained via the following iterative equation:

$$X_{n+1} = (aX_n + c) \bmod m$$

If m, a, c, and X_0 are integers, then this technique will produce a sequence of integers with each integer in the range $0 \leq X_n < m$.

The selection of values for a, c, and m is critical in developing a good random number generator. For example, consider $a = c = 1$. The sequence produced is obviously not satisfactory. Now consider the values $a = 7, c = 0, m = 32$, and $X_0 = 1$. This generates the sequence $\{7, 17, 23, 1, 7, \text{etc.}\}$, which is also clearly unsatisfactory. Of the 32 possible values, only four are used; thus, the sequence is said to have a period of 4. If, instead, we change the value of a to 5, then the sequence is $\{5, 25, 29, 17, 21, 9, 13, 1, 5, \text{etc.}\}$, which increases the period to 8.

Public-Key Cryptography
&
RSA

Terminology Related to Asymmetric Encryption

Asymmetric Keys

Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.

Public Key Certificate

A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

Public Key (Asymmetric) Cryptographic Algorithm

A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

Public Key Infrastructure (PKI)

A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

PRINCIPLES OF PUBLIC-KEY CRYPTOSYSTEMS

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

- ✓ Key distribution: Key distribution under symmetric encryption requires either
  - (1) Two communicants already share a key, which somehow has been distributed to them;
  - (2) The use of a key distribution center.
    - *Reasoned that this second requirement negated the very essence of cryptography: the ability to maintain total secrecy over your own communication.

- ✓ Digital signatures: If the use of cryptography was to become widespread, not just in military situations but for commercial and private purposes, then electronic messages and documents would need the equivalent of signatures used in paper documents. That is, could a method be devised that would stipulate, to the satisfaction of all parties, that a digital message had been sent by a particular person. This is a somewhat broader requirement than that of authentication.

Public-Key Cryptosystems

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

- ✓ It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.
- ✓ Either of the two related keys can be used for encryption, with the other used for decryption.

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 encrypted 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.

(a) Encryption with public key

- Asymmetric Key Pair can be used for: **Encryption**
  - Jim wants to send a secret message to Pam
  - Jim uses Pam's Public Key to Encrypt the message
  - Only the correlating Private Key can Decrypt

Confidentiality

(b) Encryption with private key

- Asymmetric Key Pair can be used for: **Signatures**
  - Pam wants to prove that she sent this message
  - Pam uses her Private Key to Encrypt the message
  - If Jim can decrypt with Pam's Public Key:
    - Jim knows Pam must have sent the message — Authentication
    - Jim knows message was not modified in transit — Integrity

Essential steps

- ✓ Each user generates a pair of keys to be used for the encryption and decryption of messages.
- ✓ Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others.
- ✓ If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
- ✓ When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

****With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

Conventional and Public-Key Encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if the key is kept secret. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and a related algorithm for decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if one of the keys is kept secret. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Public-Key Cryptosystem: Authentication and Secrecy

Public-key cryptosystems

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key, and the recipient decrypts the message with the recipient's private 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, which is a secret key for symmetric encryption generated for use for a particular transaction (or session) and valid for a short period of time.

Requirements for Public-Key Cryptography

1. It is computationally easy for a party B to generate a key pair (public key PU_b, private key PR_b).
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:
$$C = E(PU_b, M)$$
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:
$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$
4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.
6. The two keys can be applied in either order:
$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

What Are Digital Signatures?

- Mechanism to **determine authenticity** of a document file
- Uses **public key** cryptography mechanism
- Helpful to authenticate **long distance** official communication channels

THE RSA ALGORITHM

- ✓ Rivest-Shamir-Adleman (RSA) Algorithm
- ✓ It was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978.
- ✓ This scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.
- ✓ The RSA scheme is a cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n.
- ✓ A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}.

Key Generation by Alice

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

Encryption by Bob with Alice's Public Key

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

Decryption by Alice with Alice's Public Key

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

Two main components :

- Key Generation
- Encryption/Decryption Function

Description of the Algorithm

- ✓ 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 $\log_2(n) + 1$.
- ✓ Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.
  - $C = M^e \bmod n$
  - $M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$
- ✓ Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d.
- ✓ Thus, this is a public key encryption algorithm with a public key of $PU = \{e, n\}$ and a private key of $PR = \{d, n\}$.

Description of the Algorithm

For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of e, d, and n

   such that $M^{ed} \bmod n = M$ for all $M < n$.

2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.
3. It is infeasible to determine d given e and n.

RSA algorithm

Alice generates a public/private key pair;
Bob encrypts using Alice's public key; and Alice decrypts using her private key.

For this example, the keys were generated as follows.

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

The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$. The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

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

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

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

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

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

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

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

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

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

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

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

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

Example of RSA Algorithm

1. Choose p and q as 7 and 13 respectively, so that $n=p*q=91$

2. We can select value of e to be 5 since it satisfies $1<e<(p-1)(q-1)$

3. Value of d = $e^{-1} \bmod (p-1)(q-1) = 29$

4. Public key = (91,5), private key = (91,29)

5. Let plaintext m be 10.

Ciphertext(c) = $m^e \bmod n = 82$
Plaintext = $c^d \bmod n = 10$

A plaintext block consists of four decimal digits, or two alphanumeric characters.

KEY GENERATION

Before the application of the public-key cryptosystem, each participant must generate a pair of keys. This involves the following tasks.

- Determining two prime numbers, p and q.
- Selecting either e or d and calculating the other.

First, consider the selection of p and q. Because the value of n = pq will be known to any potential adversary, in order to prevent the discovery of p and q by exhaustive methods, these primes must be chosen from a sufficiently large set (i.e., **p and q must be large numbers**). On the other hand, the method used for finding large primes must be reasonably efficient.

CRYPTOGRAPHIC HASH FUNCTIONS

Introduction of Hash Function

- In **hash function H** accepts a variable length block of input data called as 'M' and produces the fixed size hash value can be represented as

$$h = H(M)$$

- When hash function provides security this is called ***cryptographic hash functions.***

- Hash function protects the ***integrity*** of the message. If encryption process is apply on message with hash function, it is also provide ***authentication*** and ***confidentiality***.

Figure: Block diagram of hash function

Step-1
HASH
40v3zbd5n
Sender
Receiver

Step-2
HASH
40v3zbd5n
HASH
40v3zbd5n
Compare
Sender
Receiver

Introduction of Hash Function

- A hash function provides a property that has function applied on variable amount of data (M) and then it produces the fixed amount of output data.
- If any bit or bits changes in the data, then whole hash function output data will also change.
- Cryptographic has function is ***one-way function***, which is practically ***infeasible to invert.***
- The most popular hashing algorithm is ***MD5*** and ***SHA.***

Introduction of Hash Function

❑ *Properties of hash Function*

- ***Compression:*** As per compression properties, output of the hash function is much smaller than the size of input.

- ***Pre-image resistance:*** Pre-image resistance means difficult to find the input from given hash function output. i.e., x=H(m). So if x is given, it is difficult to message m.

- ***Weak Collision Resistance:*** Given message m1, weak collision resistance means that it is difficult to produce another message m2 such that H(m1)=H(m2).

  i.e. it means it is infeasible to find two different messages with the same hash value.

- ***Strong Collision Resistance:*** Strong collision resistance means that is difficult to find any two different messages that hash to the same value.

  i.e., it means it is hard to find m1 & m2 such that same hash value H(m1)=H(m2).

****Strong And Weak Collision Resistance Are Not The Same:*** Weak collision resistance is bound to a particular input, whereas strong collision resistance applies to any two arbitrary inputs.

Introduction of Hash Function

❑ *Characteristics of hash Function*

- It is ***quick to calculate hash value (h)*** for any given message. *i.e.* $x = H(m)$.
- Hash function (H) can be applied to ***variable length of data block***.
- A ***small change*** in a message should ***change the hash value***.
- Hash function has ***one-way property***, it is impossible to generate message from given hash value.
- The hash function ***uses all the input data***.
- The hash function ***"uniformly"*** distributes the data across the entire set of possible hash values.
- The hash function generates very ***different hash values*** for ***similar message.***

Applications of Cryptographic Hash Functions

[1] Message Authentication

[2] Digital Signatures

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., there is no modification, insertion, deletion, or replay).

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 integrity 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.

Use of hash function to check data integrity

(b) Man-in-the-middle attack

Attack Against Hash Function

The hash value must be transmitted in a secure fashion.
i.e., the hash value 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.

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.

Simplified Examples of the Use of a Hash Function for Message Authentication

Simplified Examples of the Use of a Hash Function for Message Authentication

Variety of ways in which a hash code can be used to provide message authentication:

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.

What Are Digital Signatures?

- Mechanism to **determine authenticity** of a document file
- Uses **public key** cryptography mechanism
- Helpful to authenticate **long distance** official communication channels

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. Here, an attacker who wishes to alter the message would need to know the user's private key.

Simplified Examples of Digital Signatures

a) The hash code is encrypted, using public-key encryption with the sender's private key (authentication). It also provides a digital signature, because only the sender could have produced the encrypted hash code, which is the essence of the digital signature.

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.

Simple Hash Function

- Here, there are two simple hash function, all hash functions are operating using same principle.
  1. The message file is like a simple input it open a sequence on n-bit blocks.
  2. When input is processed only one block at the given time in iterative fashion to generate an n-bit hash function.

- The simple hash function is the bit-by-bit XORing done of every block.

- This can be shows the following ways:

$$C_i = B_{i1} \oplus B_{i2} \oplus \ldots \oplus B_{im}$$

where, $C_i = C_i$ is i^{th} bit if hash code, $1 <= i <= n$

$m = m$ is the number of block in the input

$B_{ij} = i^{th}$ bit in j^{th} block

$\oplus$ = XORing operation

	bit 1	bit 2	. . .	bit n
block 1	b_{11}	b_{21}		b_{n1}
block 2	b_{12}	b_{22}		b_{n2}
	⋮	⋮	⋮	⋮
block m	b_{1m}	b_{2m}		b_{nm}
hash code	C_1	C_2		C_n

- When this operation perform it produces a simple parity for each bit location this process is known as a redundancy check.

Simple Hash Function

- To increase the complexity and improve performance, use simple ways i.e. One-bit circular shift, and also the rotation on hash value after the every block is processed.

- The steps are summarized by as follows to increase complexity:
  1. Set n-bit hash value ***initially zero***.
  2. Process the each and every successive n-block of data is as follows:
     a) The ***current hash value*** is ***rotate to the left by one bit***.
     b) And performing the ***XOR operation*** in between ***block and hash value.***

Message Authentication Code (MAC)

Message Authentication Requirements

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

- ✓ **Disclosure**: Release of message contents to any person or process not possessing the appropriate cryptographic key.
- ✓ **Traffic analysis**: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
- ✓ **Masquerade:** Insertion of messages into the network from a fraudulent source. 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.
- ✓ **Content modification**: Changes to the contents of a message, including insertion, deletion, transposition, and modification.
- ✓ **Sequence modification**: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
- ✓ **Timing modification**: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
- ✓ **Source repudiation**: Denial of transmission of message by source.
- ✓ **Destination repudiation**: Denial of receipt of message by destination.

Message Authentication

- ✓ Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered.
- ✓ Message authentication may also verify sequencing and timeliness.
- ✓ A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

One of the most fascinating and complex areas of cryptography is that of message authentication and the related area of digital signatures.

- To protect message integrity (i.e. protection from modification)
- To confirm the identity of the sender

Message Authentication is concerned with

- protecting the integrity of a message
- validating identity of originator
- non-repudiation of origin (dispute resolution)

There are three types of functions that may be used to produce an authenticator:

- ✓ A hash function
- ✓ Message encryption
- ✓ Message authentication code (MAC)

Basic Uses of Message Encryption

Communication without authentication

Eve can simply change the message

Shared key *k* to generate authenticate message

Integrity Protection with MAC

Eve can not forge MAC when k is unknown

Shared key *k* to generate authenticate message

MAC Authentication (I)

- MAC allows **two** or more mutually trusting parties to authenticate messages sent between members

MAC Authentication (II)

- MAC allows two or **more** mutually trusting parties to authenticate messages sent between members

Integrity with Hash

Can we simply send the hash with the message to serve message authentication ?

Ans: No, Eve can change the message and recompute the hash.

Using hash needs more appropriate procedure to guarantee integrity

Message Authentication Code

- ✓ 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.
- ✓ 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. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.

  The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

  If the message includes a sequence number (such as is used with HDLC, X.25,and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.
- ✓ A MAC function is like encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function.

(a) Message authentication

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

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

Basic Uses of Message Authentication code (MAC)

MAC Properties

- A MAC is a cryptographic checksum

$$MAC = C_K(M)$$

  condenses a variable-length message M
  using a secret key K
  to a fixed-sized authenticator

- A many-to-one function
  potentially many messages have same MAC

1. There are several applications in which the same message is broadcast to several destinations. Examples are notification to users that the network is now unavailable or an alarm signal in a military control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated message authentication code. The responsible system has the secret key and performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.

2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis; messages being chosen at random for checking.

3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program.

4. For some applications, it may not be of concern to keep messages secret, but it is important to authenticate messages.

5. Separation of authentication and confidentiality functions affords architectural flexibility. For example, it may be desired to perform authentication at the application level but to provide confidentiality at a lower level, such as the transport layer.

6. A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents. With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

Finally, note that the ***MAC does not provide a digital signature, because both sender and receiver share the same key***.

SYMMETRIC KEY DISTRIBUTION USING SYMMETRIC ENCRYPTION

Key Distribution Technique

- It is defined as a means of delivering a key to two parties who wish to exchange data without allowing others to see the key.
- The strength of any cryptographic system rests with the key distribution technique.
- For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others.
- Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.

Key Distribution Technique

For two parties A and B, key distribution can be achieved in several ways, as follows:

- 1) A can select a key and physically deliver it to B.
- 2) A third party can select the key and physically deliver it to A and B.
- 3) If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
- 4) If A and B each has an encrypted connection to a third-party C, C can deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link. However, for **end-to-end encryption** over a network, manual delivery is awkward.

In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device needs several keys supplied dynamically. The problem is especially difficult in a wide-area distributed system.

Key Distribution Technique

The Use of a Key Hierarchy

For end-to-end encryption, some variation on option 4 has been widely adopted.

In this scheme, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.

Each user must share a unique key with the key distribution center for purposes of key distribution.

The use of a key distribution center is based on the use of a hierarchy of keys.

At a minimum, two levels of keys are used. Communication between end systems is encrypted using a temporary key, often referred to as a session key.

Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded. Each session

key is obtained from the key distribution center over the same networking facilities used for end-user communication. Accordingly, session keys are transmitted in encrypted form, using a master key that is shared by the key distribution center and an end system or user.

Key Distribution Technique

The Use of a Key Hierarchy

For each end system or user, there is a unique master key that it shares with the key distribution center. These master keys must be securely distributed in some fashion. However, the scale of the problem is vastly reduced.

If there are N entities that wish to communicate in pairs, then, as many as [N(N - 1)]/2 session keys are needed at any one time. However, only N master keys are required, one for each entity.

Thus, master keys can be distributed in some non-cryptographic way, such as physical delivery.

A Key Distribution Scenario

- Here, each user shares a unique master key with the key distribution center (KDC). Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.

A has a master key *Ka*, known only to itself and the KDC; similarly, B shares the master key *Kb* with the KDC. The following steps occur.

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, *N*1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

A Key Distribution Scenario

2. The KDC responds with a message encrypted using *Ka*. Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

- The one-time session key, *Ks*, to be used for the session
- The original request message, including the nonce, to enable A to match this response with the appropriate request

Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.

In addition, the message includes two items intended for B:

- The one-time session key, *Ks*, to be used for the session
- An identifier of A (e.g., its network address), *IDA*

These last two items are encrypted with *Kb* (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

A Key Distribution Scenario

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B. Because this information is encrypted with Kb, it is protected from eavesdropping. B now knows the session key (Ks), knows that the other party is A (from IDA), and knows that the information originated at the KDC (because it is encrypted using Kb). At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N2, to A.

5. Also, using Ks, A responds with f(N2), where f is a function that performs some transformation on N2 (e.g., adding one).

User Authentication

REMOTE USER-AUTHENTICATION PRINCIPLES

USER-AUTHENTICATION PRINCIPLES

- User authentication is the fundamental building block and the primary line of defense.
- User authentication is the basis for most types of access control and for user accountability.
- RFC4949 (*Internet Security Glossary*) defines user authentication as the process of verifying an identity claimed by or for a system entity.

This process consists of two steps:

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

USER-AUTHENTICATION PRINCIPLES

- For example, user ALICE ANDERSON could have the user identifier ALICEANDER . This information needs to be stored on any server or computer system that Alice wishes to use and could be known to system administrators and other users.
- 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 email, but because her password is secret, no one can pretend to be Alice.

The NIST Model for Electronic User Authentication

- NIST SP 800-63-2 (*Electronic Authentication Guideline*, August 2013) defines electronic user authentication as the process of establishing confidence in user identities that are presented electronically to an information system.
- Systems can use the authenticated identity to determine if the authenticated individual is authorized to perform particular functions, such as database transactions or access to system resources.
- In many cases, the authentication and transaction or other authorized function takes place across an open network such as the Internet. Equally authentication and subsequent authorization can take place locally, such as across a local area network.
- SP 800-63-2 defines a general model for user authentication that involves a number of entities and procedures.

The NIST Model for Electronic User Authentication

The NIST SP 800-63-2 E-Authentication Architectural Model

The NIST Model for Electronic User Authentication

- The initial requirement for performing user authentication is that the user must be registered with the system.

The following is a typical sequence for registration.

- ✓ An applicant applies to a registration authority (RA) to become a subscriber of a credential service provider (CSP). In this model, the RA is a trusted entity that establishes and vouches for the identity of an applicant to a CSP.
- ✓ The CSP then engages in an exchange with the subscriber. Depending on the details of the overall authentication system, the CSP issues some sort of electronic credential to the subscriber.
- ✓ The credential is a data structure that authoritatively binds an identity and additional attributes to a token possessed by a subscriber, and can be verified when presented to the verifier in an authentication transaction.
- ✓ The token could be an encryption key or an encrypted password that identifies the subscriber. The token may be issued by the CSP, generated directly by the subscriber, or provided by a third party. The token and credential may be used in subsequent authentication events.

The NIST Model for Electronic User Authentication

- Once a user is registered as a subscriber, the actual authentication process can take place between the subscriber and one or more systems that perform authentication and, subsequently, authorization.
- The party to be authenticated is called a claimant and the party verifying that identity is called a verifier.
- When a claimant successfully demonstrates possession and control of a token to a verifier through an authentication protocol, the verifier can verify that the claimant is the subscriber named in the corresponding credential.
- The verifier passes on an assertion about the identity of the subscriber to the relying party (RP). That assertion includes identity information about a subscriber, such as the subscriber name, an identifier assigned at registration, or other subscriber attributes that were verified in the registration process.
- The RP can use the authenticated information provided by the verifier to make access control or authorization decisions.

This model illustrates the key roles and functions needed for a secure authentication system.

Means of 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 key cards, 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.

Mutual Authentication

Mutual authentication protocols 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 & 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: The second issue, 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.

Chapter 17: Transport Level Security

Introduction to TLS

- TLS is a widely adopted security protocol which helps in maintaining privacy and data security for communications over internet.

- The important use of TLS is to encrypt the communication between web application and the server.

- TLS is an advanced version of SSL (secure sockets layer). HTTPS is an implementation of the TLS encryption developed from the HTTP protocol which is used by all websites and web services.

Web Security Considerations

- The world wide web (www) is fundamentally a server application running over TCP/IP protocol.
- Web browsers although easy to use but have many security problems due to which the attacker can gain access to data and systems connected over the server.
- There exist 2 types of web security threats: Passive and active. In passive threat, the communication between browser and server is only eavesdropped but in active threats, the messages communicated between browser and server are altered.

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

TLS Architecture

- TLS is designed to make use of TCP protocol for end to end secure service communication. TLS is not a single protocol but contains different layers.
- The HTTP part provides the transfer service for client server interaction and other layers are a part of the handshake protocol.
- The different layers are the handshake protocol, the change cipher spec protocol and the alert protocol.
- TCP/IP is the basic layer of any network protocol which helps in maintaining communication between client and server.

Handshake protocol
Change cipher spec protocol
Alert protocol
HTTP
Heartbeat protocol
Record protocol
TCP
IP

- Record protocol provides 2 services for TLS connection i.e confidentiality and message integrity. The handshake protocol defines a secret key for maintaining integrity and confidentiality using the concept of MAC.
- The change cipher spec protocol consists of a single message of 1 byte and the purpose is to cause the pending state to be copied into the current state.
- The alert protocol helps in conveying all the alert messages related to TLS to the client. The heartbeat protocol is typically used to monitor the availability of the protocol entity.
- HTTPS refers to secure communication between browser and server.

TLS Record Protocol Operation

- The TLS Record Protocol provides two services for TLS connections:
- ■ **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of TLS payloads.
- ■ **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure 17.3 TLS Record Protocol Operation

Steps in Record Protocol

- Take the application message to be transmitted.
- Then fragment the message, each upper layer is fragmented into 16,384 bytes or less.
- Next compression step is there which is optionally applied which is lossless compression. It may not increase the content length by more than 1024 bytes.
- Next step is computation of MAC over the compressed data, HMAC algorithm is used for this.
- Next the compressed message plus MAC are encrypted symmetrically and a header is prepended to the encrypted message.
- Resulting unit is transmitted over the TCP segment.

HMAC Algorithm (hashing based MAC)

$$\mathrm{HMAC}_K(M) = \mathrm{H}[(K^+ \oplus \mathrm{opad}) \parallel \mathrm{H}[(K^+ \oplus \mathrm{ipad}) \parallel M]]$$

where

H = embedded hash function (for TLS, either MD5 or SHA-1)

M = message input to HMAC

K^+ = secret key padded with zeros on the left so that the result is equal to the block length of the hash code (for MD5 and SHA-1, block length = 512 bits)

ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)

opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

For TLS, the MAC calculation encompasses the fields indicated in the following expression:

HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type || TLSCompressed.version || TLSCompressed.length || TLSCompressed.fragment)

Handshake Protocol

- The most complex part of TLS 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 a TLS 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. All of these have the format. Each message has three fields:
- ■ **Type (1 byte):** Indicates one of 10 protocol messages as in next slide..
- ■ **Length (3 bytes):** The length of the message in bytes.
- ■ **Content (# 0 bytes):** The parameters associated with this message.

Phases of Handshake Protocol Action

Phase 1 initiates a logical connection and establishes the security capabilities that will be associated with it. The exchange is initiated by the client, which sends a **client_hello message** with the following parameters:

- **Version:** The highest TLS version understood by the client.
- **Random:** A client-generated random structure consisting of a 32-bit time stamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during key exchange to prevent replay attacks.
- **Session ID:** A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or to create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.

Table 17.2 TLS Handshake Protocol Message Types

Message Type	Parameters
hello_request	null
client_hello	version, random, session id, cipher suite, compression method
server_hello	version, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificates
server_key_exchange	parameters, signature
certificate_request	type, authorities
server_done	null
certificate_verify	signature
client_key_exchange	parameters, signature
finished	hash value

- ■ **CipherSuite:** This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a CipherSpec; these are discussed subsequently.
- ■ **Compression Method:** This is a list of the compression methods the client supports.

Phase 2: Server may send certificate, key exchange and request certificate. Server signals the end of Hello message phase.

Phase 3: Client sends certificate if requested. Client sends key exchange and may send certificate verification.

Phase 4: Change cipher suit and finish handshake protocol.

Figure 17.6 Handshake Protocol Action

TLS Components

- The TLS protocol accomplishes these three components:
  - Encryption
  - Authentication
  - Integrity

TLS concepts

- TLS Session : It 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. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.

- TLS Connection: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For TLS, such connections are peer-to peer relationships. The connections are transient. Every connection is associated with one session.

Chapter 20: IP Security

Introduction

- It includes the security protocol across IP network for maintaining message integrity and confidentiality.

- IP security protocol is termed as IPsec and is used to encrypt the application layered data for providing security to routers over public networks.

- Ipsec services are access control, connectionless integrity, data origin authentication and limited traffic flow confidentiality.

Applications of IPsec

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.

- **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.

- **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. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer.

Benefits of IPsec

- IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.
- 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. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
- IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.
- When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

Ipsec Architecture

(a) Tunnel-mode format

Ipsec Documents

- AH here is the authentication header which provides the message authentication.
- ESP is the encapsulating security payload which helps in providing encryption authentication.
- Internet Key exchange describing the key management schemes for use in IPsec.
- Cryptographic algorithms used for encryption, message authentication, pseudorandom functions and cryptographic key exchanges.
- Others: All the other IPsec related security management and management information content.