SUDDHANANDA SCHOOL OF MANAGEMENT AND COMPUTER SCIENCE

A
LECTURE NOTES
ON

NETWORK SECURITY

MCA 2ND YEAR
(4TH SEMESTER)

By

Sujata Aparajita
(ASSISTANT PROFESSOR)
DEPARTMENT OF MCA

Chapter 1

UNIT 1: **INTRODUCTION** Services, Mechanisms and attacks - The OSI Security Architecture- A Model for Network Security – Classical Encryption Technique – Symmetric Cipher Model – Substitution Technique – Rotor Machines – Steganograph

COURSE OBJECTIVES

- To understand the fundamentals of Cryptography.
- To acquire knowledge on standard algorithms used to provide confidentiality, integrity and authenticity.
- To explore the various key distribution and management schemes.
- To understand how to deploy encryption techniques to secure data in transit across data networks.
- To learn various mechanisms for network security to protect against the threats in the networks.

COURSE OUTCOMES

On completion of the course, student will be able to

CO1 - Implement various symmetric encryption techniques for given applications.

CO2 - Illustrate various public key encryption techniques.

CO3 - Understand various key encryption mechanisms and key management strategies that can be applied for real time transactions.

CO4 - Evaluate authentication and hash algorithms.

CO5 - Summarize the basic network security mechanisms.

CO6 - Basic concepts of system level security.

Security Attack, Services and Mechanism

Introduction:

With the introduction of the computer, the need for automatic tools for protecting files and other information stored on the computer became evident. This is especially the case for a shared system, such as a time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone network, data network, or the Internet. The generic name for the collection of tools designed to protect data and to thwart hackers is **computer security**. The second major change that affected security is the introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer.

Network security measures are needed to protect data during their transmission. internet security, which consists of measures to deter, prevent, detect, and correct security violations that involve the transmission of information. Consider the following example of security violations: User A transmits a file to user B. The file contains sensitive information (e.g., payroll records) that is to be protected from disclosure. User C, who is not authorized to read the file, is able to monitor the transmission and capture a copy of the file during its transmission.

1.1 The OSI Security Architecture

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. ITU-T Recommendation X.800, **Security Architecture for OSI**, defines such a systematic approach. The OSI security architecture is useful to managers as a way of organizing the task of providing security.

The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended

to counter security attacks, and they make use of one or more security mechanisms to provide the service.

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

1.2 Security Attacks

A useful means of classifying security attacks, is in terms of **passive attacks and active attacks**. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks:

Passive attacks are in the nature of **eavesdropping on, or monitoring of, transmissions**. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are **release of message contents and traffic analysis.**

- The **release of message contents** is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.
- A second type of passive attack, **traffic analysis**, is subtler. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, **the emphasis in dealing with passive attacks is on prevention rather than detection.**

Active Attacks:

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

1. **masquerade,**
2. **replay,**
3. **modification of messages, and**
4. **denial of service.**

- A **masquerade** takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.
- **Replay** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- **Modification of messages** simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect.
- **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.

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

or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention. Figure 1 shows the passive and active attack types.

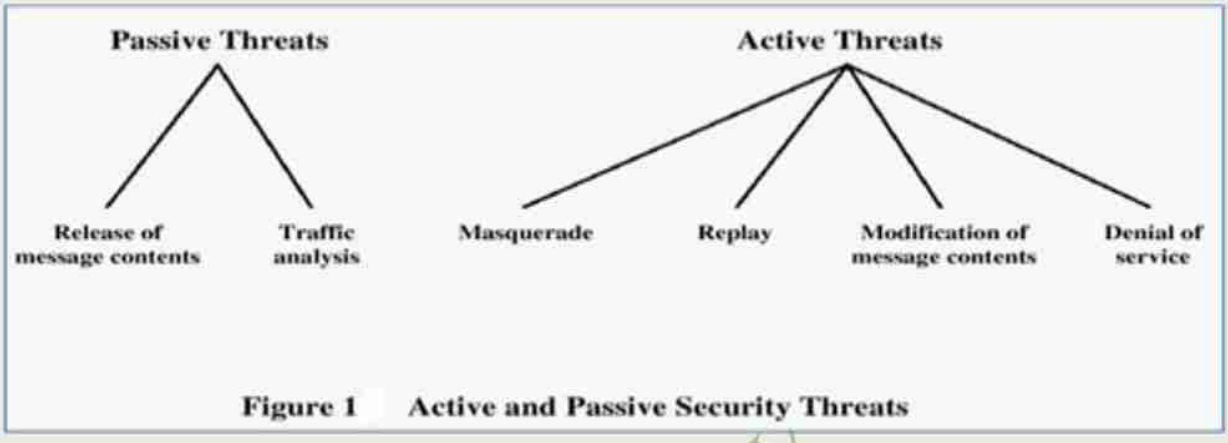


Figure 1 Active and Passive Security Threats

Figure1: Active and passive threats

1.3 Security Services

X.800 defines a security service as a service provided by a protocol layer of communicating open systems, which ensures adequate security of the systems or of data transfers. Perhaps a clearer definition is: a processing or communication service that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanisms.

1.3.1 Authentication

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception. Two specific authentication services are defined in X.800:

- **Peer entity authentication:** Provides for the corroboration of the identity of a peer entity in an association. It is provided for use at the establishment of, or at times during the data transfer

phase of, a connection. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.

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

1.3.2 Access Control

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

1.3.3 Data Confidentiality

Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are less useful than the broad approach and may even be more complex and expensive to implement.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

1.3.4 Data Integrity

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection. A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. On the other hand, a connectionless integrity service, one that deals with individual messages without regard to any larger context, generally provides protection against message modification only.

We can make a distinction between the service with and without recovery. Because the integrity service relates to active attacks, we are concerned with detection rather than prevention. If a violation of integrity is detected, then the service may simply report this violation, and some other portion of software or human intervention is required to recover from the violation. Alternatively, there are mechanisms available to recover from the loss of integrity of data, as we

will review subsequently. The incorporation of automated recovery mechanisms is, in general, the more attractive alternative.

1.3.5 Nonrepudiation

Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

1.4 Security Mechanisms

Following is the list of the security mechanisms defined in X.800. As can be seen the mechanisms are divided into those that are implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service.

X.800 distinguishes between reversible encipherment mechanisms and irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted. Irreversible encipherment mechanisms include hash algorithms and message authentication codes, which are used in digital signature and message authentication applications.

A. Specific Security Mechanisms

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

1. **Encipherment:** The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.
2. **Digital Signature:** Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).
3. **Access Control:** A variety of mechanisms that enforce access rights to resources.
4. **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
5. **Authentication Exchange:** A mechanism intended to ensure the identity of an entity by means of information exchange.
6. **Traffic Padding:** The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

7. **Routing Control:** Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.
8. **Notarization:** The use of a trusted third party to assure certain properties of a data exchange.

B. <u>Pervasive Security Mechanisms</u>

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

1. **Trusted Functionality:** That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
2. **Security Label:** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource. **3. Event Detection:** Detection of security-relevant events.
4. **Security Audit Trail:** Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.
5. **Security Recovery:** Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

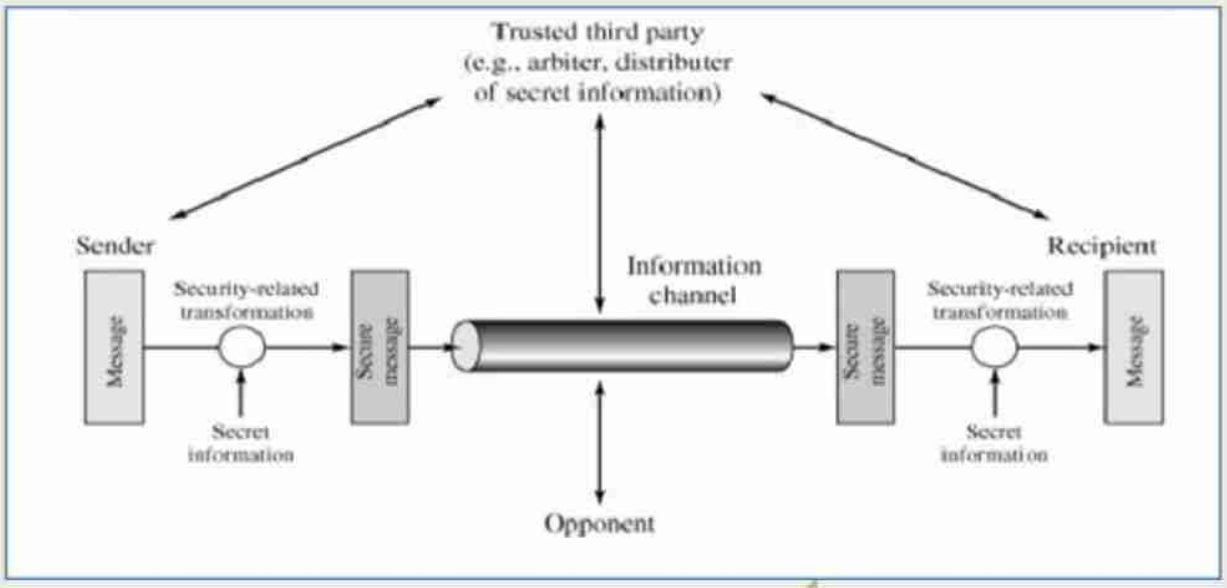
1.5 A Model for Network Security

A model for much of what we will be discussing is captured, in very general terms, in Figure 2. A message is to be transferred from one party to another across some sort of internet. 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.

Figure 2: Model for network security



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

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

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

The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. Or, 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). 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 t**hat 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.

1.6 CLASSICAL ENCRYPTION TECHNIQUES

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of public-key encryption in the 1970s. It remains by far the most widely used of the two types of encryption. Part One examines a number of symmetric ciphers. In this chapter, we begin with a look at a general model for the symmetric encryption process; this will enable us to understand the context within which the algorithms are used. Next, we examine a variety of algorithms in use before the computer era. Finally, we look briefly at a different approach known as steganography. Chapter 3 examines the most widely used symmetric cipher: DES.

Before beginning, we define some terms. An original message is known as the **plaintext**, while the coded message is called the **ciphertext**. The process of converting from plaintext to ciphertext is known as **enciphering** or **encryption**; restoring the plaintext from the ciphertext is **deciphering** or **decryption**. The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system** or a **cipher**. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls "breaking the code." The areas of cryptography and cryptanalysis together are called **cryptology**.

1.7 Symmetric Cipher Model

A symmetric encryption scheme has five ingredients (Figure):

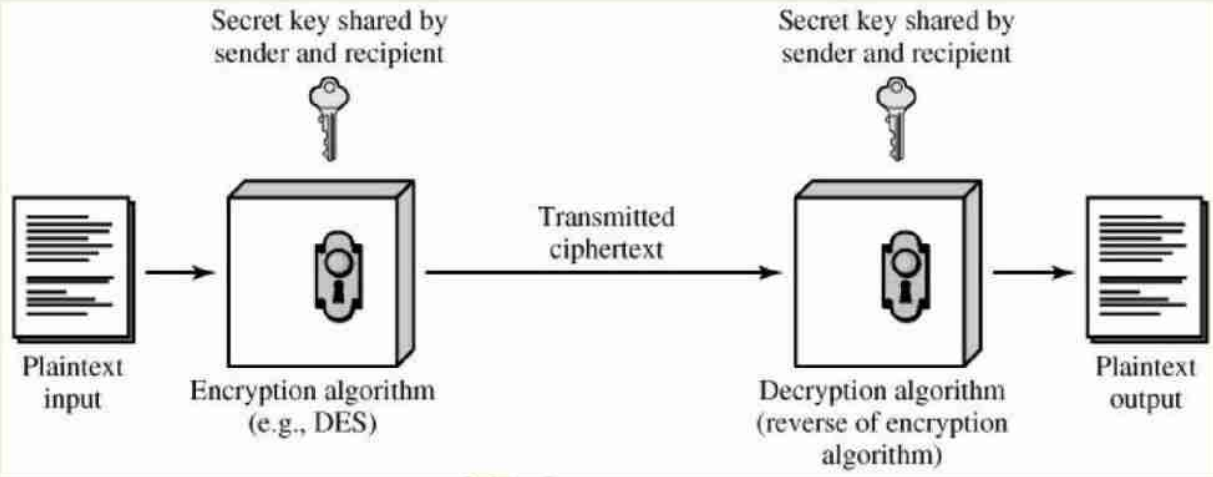
Plaintext:

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.

Cipher text: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts. The cipher text 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 cipher text and the secret key and produces the original plaintext.

Figure Simplified Model of Conventional Encryption



There are two requirements for secure use of conventional encryption:

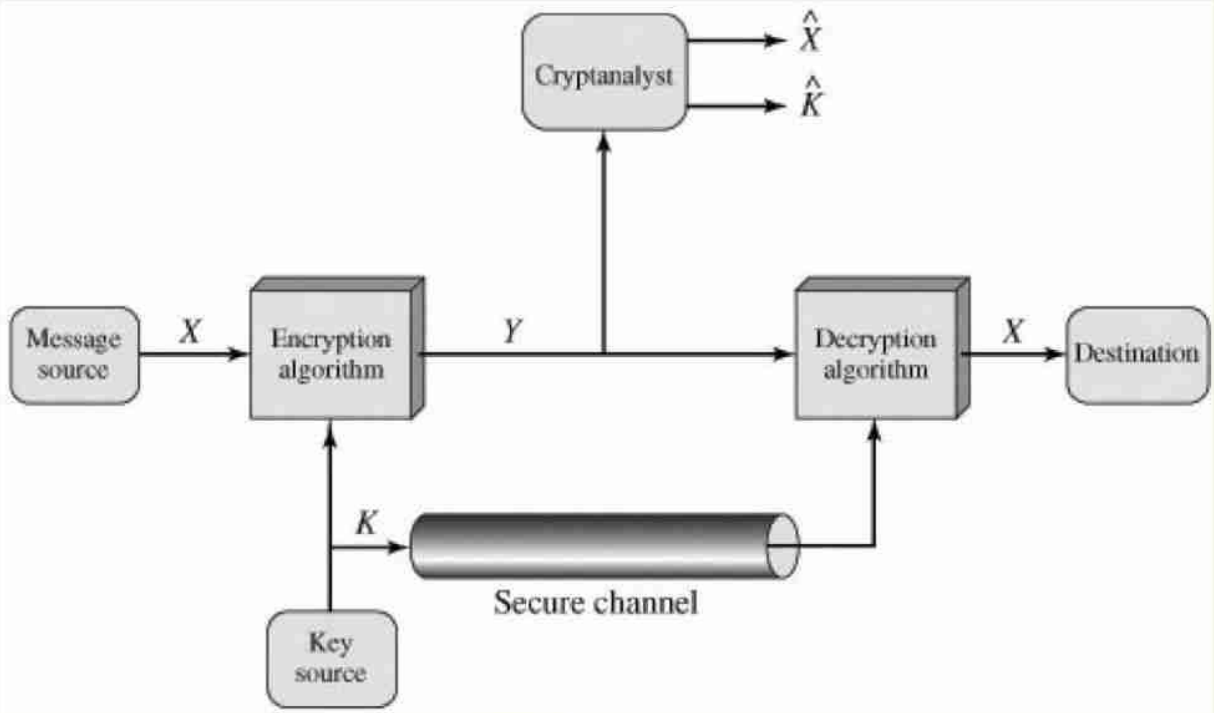
1. We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The 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.

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

We assume that it is impractical to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption is what makes it feasible for widespread use. The fact that the algorithm need not be kept secret means that manufacturers can and have developed low-cost chip implementations of data encryption algorithms. These chips are widely available and incorporated into a number of products. With the use of symmetric encryption, the principal security problem is maintaining the secrecy of the key.

Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure 2.2. A source produces a message in plaintext, X = [X1, X2, ..., XM]. The M elements of X are letters in

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



Model of Conventional Cryptosystem

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

This notation indicates thatY is produced by using encryption algorithm E as a function of the plaintexXt , with the specific function determined by the value of the key K.

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

X = D(K, Y)

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate . Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate .

Cryptography

Cryptographic systems are characterized along three independent dimensions:

1. **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: substitution, in which each element in the plaintext (bit,

letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.

2. **The number of keys used.** If both sender and receiver use the same key, the system is referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.

3. **The way in which the plaintext is processed.** A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

Cryptanalysis

Typically, the objective of attacking an encryption system is to recover the key in use rather then 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. If either type of attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.

Table summarizes the various types of **cryptanalytic attacks**, based on the amount of information known to the cryptanalyst. The most difficult problem is presented when all that is available is the cipher text only. In some cases, not even the encryption algorithm is known, but in general we can assume that the opponent does know the algorithm used for encryption. One possible attack under these circumstances is the brute-force approach of trying all possible keys. If the key space is very large, this becomes impractical. Thus, the opponent must rely on an analysis of the cipher text itself, generally applying various statistical tests to it.

Table Types of Attacks on Encrypted Messages

Type of Attack	**Known to Cryptanalyst**
Cipher text only	• Encryption algorithm • Cipher text
Known plaintext	• Encryption algorithm • Cipher text

- One or more plaintext-cipher text pairs formed with the secret key **Chosen plaintext**

- Encryption algorithm
- Cipher text
- Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key **Chosen cipher text**
- Encryption algorithm
- Cipher text

- Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key **Chosen text**
- Encryption algorithm
- Ciphertext
- Plaintext message chosen by cryptanalyst, together with its corresponding
- ciphertext generated with the secret key
- Purported ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

A **brute-force attack** involves trying every possible key until an intelligible translation of the ciphertext into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.
Results are shown for four binary key sizes. The 56-bit key size is used with the DES (Data Encryption Standard) algorithm, and the 168-bit key size is used for triple DES. The minimum key size specified for AES (Advanced Encryption Standard) is 128 bits. Results are also shown for what are called substitution codes that use a 26-character key (discussed later), in which all possible permutations of the 26 characters serve as keys. For each key size, the results are shown assuming that it takes 1 ms to perform a single decryption, which is a reasonable order of magnitude for today's machines. With the use of massively parallel organizations of microprocessors, it may be possible to achieve processing rates many orders of magnitude greater.

1.8 Substitution Techniques

In this section and the next, we examine a sampling of what might be called classical encryption techniques. A study of these techniques enables us to illustrate the basic approaches to symmetric encryption used today and the types of cryptanalytic attacks that must be anticipated. The two basic building blocks of all encryption techniques are substitution and transposition. We examine these in the next two sections. Finally, we discuss a system that combines both substitution and transposition. A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with ciphertext bit patterns

1.8.1 Caesar Cipher

The earliest known use of a substitution cipher, and the simplest, was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example, plain: meet me after the toga party
Cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows: plain: a b c d e f g h i j k l m n o p q r s t u v w x y z cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C Let us assign a numerical equivalent to each letter:

a b c d e f g h i j k l m 0 1 2 3 4
5 6 7 8 9 10 11 12 n o p q r s t
u v w x y z
13 14 15 16 17 18 19 20 21 22 23 24 25

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

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

A shift may be of any amount, so that the general Caesar algorithm is $C = E(k, p) = (p + k) \bmod 26$ where k takes on a value in the range 1 to 25. The decryption algorithm is simply $p = D(k, C) = (C\ k) \bmod 26$

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys. Figure 2.3 shows the results of applying this strategy to the example ciphertext. In this case, the plaintext leaps out as occupying the third line.

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

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

1. The encryption and decryption algorithms are known.

2. There are only 25 keys to try.

3. The language of the plaintext is known and easily recognizable

In most networking situations, we can assume that the algorithms are known. What generally makes brute-force cryptanalysis impractical is the use of an algorithm that employs a large number of keys. For example, the triple DES algorithm, examined in Chapter 6, makes use of a 168-bit key, giving a key space of 2 168 or greater than 3.7 x 10^5 possible keys. The third characteristic is also significant. If the language of the plaintext is unknown, then plaintext output may not be recognizable. Furthermore, the input may be abbreviated or compressed in some fashion, again making recognition difficult.

1.8.2 Monoalphabetic Ciphers

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Recall the assignment for the Caesar cipher:
plain: a b c d e f g h i j k l m n o p q r s t u v w x y z cipher: D E F G H
I J K L M N O P Q R S T U V W X Y Z A B C
If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4 x 10^26 possible keys. This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis. Such an approach is referred to as a **monoalphabetic substitution cipher**, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message. There is, however, another line of attack. If the cryptanalyst knows the nature of the plaintext (e.g., non compressed English text), then the analyst can exploit the regularities of the language. To see how such a cryptanalysis might proceed. The ciphertext to be solved is

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ

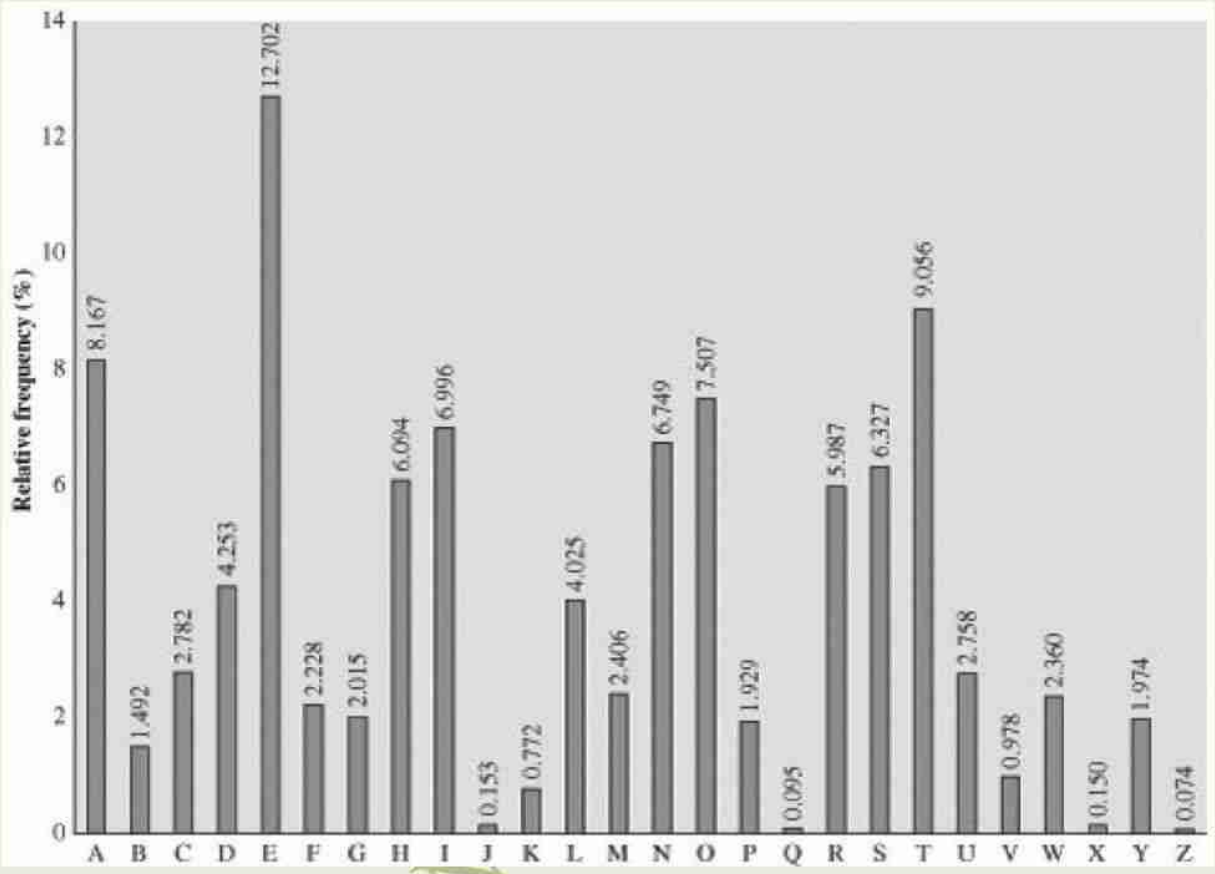
VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX

EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English, such as is shown in Figure 2.5. If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match. In any case, the relative frequencies of the letters in the ciphertext (in percentages) are as follows:

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

Figure 2.5. Relative Frequency of Letters in English Text



Comparing this breakdown with Figure 2.5, it seems likely that cipher letters P and Z are the equivalents of plain letters e and t, but it is not certain which is which. The letters S, U, O, M, and H are all of relatively high frequency and probably correspond to plain letters from the set {a, h, i, n, o, r, s}.The letters with the lowest frequencies (namely, A, B, G, Y, I, J) are likely included in the set {b, j, k, q, v, x, z}. There are a number of ways to proceed at this point. We could make some tentative assignments and start to fill in the plaintext to see if it looks like a reasonable "skeleton" of a message. A more systematic approach is to look for other regularities. For example, certain words may be known to be in the text. Or we could look for repeating sequences of cipher letters and try to deduce their plaintext equivalents.

A powerful tool is to look at the frequency of two-letter combinations, known as digrams. A table similar to Figure 2.5 could be drawn up showing the relative frequency of digrams. The most common such digram is th. In our cipher text, the most common digram is ZW, which appears three times. So we make the correspondence of Z with t and W with h. Then, by our earlier hypothesis, we can equate P with e. Now notice that the sequence ZWP appears in the cipher text, and we can translate that sequence as "the." This is the most frequent trigram (threeletter combination) in English, which seems to indicate that we are on the right track. Next, notice the sequence ZWSZ in the first line. We do not

know that these four letters form a complete word, but if they do, it is of the form th_t. If so, S equates with a. So far, then, we have

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

Only four letters have been identified, but already we have quite a bit of the message. Continued analysis of frequencies plus trial and error should easily yield a solution from this point. The complete plaintext, with spaces added between words, follows: it was disclosed yesterday that several informal but direct contacts have been made with political representatives of the viet cong in Moscow

Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet. A countermeasure is to provide multiple substitutes, known as homophones, for a single letter. For example, the letter e could be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone used in rotation, or randomly. If the number of symbols assigned to each letter is proportional to the relative frequency of that letter, then single-letter frequency information is completely obliterated. The great mathematician Carl Friedrich Gauss believed that he had devised an unbreakable cipher using homophones. However, even with homophones, each element of plaintext affects only one element of ciphertext, and multiple-letter patterns (e.g., digram frequencies) still survive in the ciphertext, making cryptanalysis relatively straightforward. Two principal methods are used in substitution ciphers to lessen the extent to which the structure of the plaintext survives in the ciphertext: One approach is to encrypt multiple letters of plaintext, and the other is to use multiple cipher alphabets. We briefly examine each.

1.8.3 Playfair Cipher

The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams.

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers's Have His Carcase.

M O N A R

C H Y B D

E F G I/J K

L P Q S T

U V W X Z

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

1. Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

2. Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.

3. Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.

4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 x 26 = 676 digrams, so that identification of individual digrams is more difficult. Furthermore, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis much more difficult. For these reasons, the Playfair cipher was for a long time considered unbreakable. It was used as the standard field system by the British Army in World War I and still enjoyed considerable use by the 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.

1.8.4 Hill Cipher

Another interesting multilettered cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. The encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters. The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1 ... z = 25). For m = 3, the system can be described as follows:
c1 = (k11P1 + k12P2 + k13P3) mod 26 c2 =
(k21P1 + k22P2 + k23P3) mod 26 c3 =
(k31P1 + k32P2 + k33P3) mod 26
This can be expressed in term of column vectors and matrices:

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

or

C = **KP** mod 26 where **C** and **P** are column vectors of length 3, representing the plaintext and ciphertext, and **K** is a 3 x 3 matrix, representing the encryption key. Operations are performed mod 26. For example, consider the plaintext "paymoremoney" and use the encryption key

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

The first three letters of the plaintext are represented by the vector

$\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$. Then $\mathbf{K}\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix}$ = LNS. Continuing in this fashion, the ciphertext for the entire plaintext is LNSHDLEWMTRW.

Decryption requires using the inverse of the matrix **K**. The inverse **K-**1 of a matrix **K** is defined by the equation **KK-**1 = **K-**1**K** = **I**, where **I** is the matrix that is all zeros except for ones along the main diagonal from upper left to lower right. The inverse of a matrix does not always exist, but when it does, it satisfies the preceding equation. In this case, the inverse

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

This is demonstrated as follows:

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}\begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

It is easily seen that if the matrix **K-**1 is applied to the ciphertext, then the plaintext is recovered. To explain how the inverse of a matrix is determined, we make an exceedingly brief excursion into linear algebra. For any square matrix (m x m) the **determinant** equals the sum of all the products that can be formed by taking exactly one element from each row and exactly one element from each column, with certain of the product terms preceded by a minus sign. For a
2*2 matrix,

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

the determinant is k11k22 k12k21. For a 3 x 3 matrix, the value of the determinant isk 11k22k33

+ k21k32k13 + k31k12k23 k31k22k13 k21k12k33 k11k32k23. If a square matrix**A** has a nonzero determinant, then the inverse of the matrix is computed asA^1ij =$(1)^{I+J}$(Dij)/ded(**A**), where (Dij) is the subdeterminant formed by deleting the ith row and the jth column of **A** and det(**A**) is the determinant of **A**. For our purposes, all arithmetic is done mod 26.

In general terms, the Hill system can be expressed as follows:

C = E(**K, P**) = **KP** mod 26

P = D(**K, P**) = K^1**C** mod 26 = K^1**KP** = **P**

As with Playfair, the strength of the Hill cipher is that it completely hides single-letter frequencies. Indeed, with Hill, the use of a larger matrix hides more frequency information. Thus a 3 x 3 Hill cipher hides not only single-letter but also two-letter frequency information.

Although the Hill cipher is strong against a cipher text-only attack, it is easily broken with a known plaintext attack. For an m x m Hill cipher, suppose we have m plaintext-ciphe r text pairs, each of length m. We label the pairs

$$\mathbf{P}_j = \begin{pmatrix} p_{1j} \\ p_{2j} \\ \vdots \\ p_{mj} \end{pmatrix} \text{ and } \mathbf{C}_j = \begin{pmatrix} c_{1j} \\ c_{2j} \\ \vdots \\ c_{mj} \end{pmatrix} \text{ such that } \mathbf{C}_j = \mathbf{K}\mathbf{P}_j \text{ for } 1 \le j \le m \text{ and for some}$$

unknown key matrix **K**. Now define two m x m matrices **X** = (Pij) and **Y** = (Cij). Then we can form the matrix equation **Y** = **KX**. If **X** has an inverse, then we can determine **K** = **YX**1.If **X** is not invertible, then a new version of **X** can be formed with additional plaintext-ciphertext pairs until an invertible **X** is obtained. Suppose that the plaintext "friday" is encrypted using a 2 x 2

Hill cipher to yield the ciphertext PQCFKU. Thus, we know that

$$\mathbf{K}\begin{pmatrix} 5 \\ 17 \end{pmatrix} \bmod 26 = \begin{pmatrix} 15 \\ 16 \end{pmatrix}; \mathbf{K}\begin{pmatrix} 8 \\ 3 \end{pmatrix} \bmod 26 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}; \quad \text{and} \quad \mathbf{K}\begin{pmatrix} 0 \\ 24 \end{pmatrix} \bmod 26 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

Using the first two plaintext-ciphertext pairs, we have

$$\begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \mathbf{K}\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \bmod 26$$

The inverse of **X** can be computed:

$$\mathbf{K} = \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix}\begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} = \begin{pmatrix} 137 & 60 \\ 149 & 107 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

Polyalphabetic Ciphers

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**. All these techniques have the following features in common:

1. A set of related monoalphabetic substitution rules is used.

2. A key determines which particular rule is chosen for a given transformation.

The best known, and one of the simplest, such algorithm is referred to as the Vigenère cipher. In this scheme, the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers, with shifts of 0 through 25. Each cipher is denoted by a key letter,Mwhich is the ciphertext letter that substitutes for the plaintext letter a. Thus, a Caesar cipher with a shift of 3 is denoted by the key value d. To aid in understanding the scheme and to aid in its use, a matrix known as the Vigenère tableau is constructed (Table 2.3). Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of encryption is simple: Given a key letter x and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled x and the column labeled y; in this case the ciphertext is V.

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

Table 2.3. The Modern Vigenère Tableau

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is deceptive, the message "we are discovered save yourself" is encrypted as follows: key: deceptivedeceptivedeceptive plaintext: wearediscoveredsaveyourself ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Decryption is equally simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column. The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus, the letter frequency information is obscured. However, not all knowledge of the plaintext structure is lost. For example, Figure 2.6 shows the frequency distribution for a Vigenère cipher with a keyword of length 9. An improvement is achieved over the Playfair cipher, but considerable frequency information remains.

It is instructive to sketch a method of breaking this cipher, because the method reveals some of the mathematical principles that apply in cryptanalysis. First, suppose that the opponent believes that the ciphertext was encrypted using either monoalphabetic substitution or a Vigenère cipher. A simple test can be made to make a determination. If a monoalphabetic substitution is used, then the statistical properties of the ciphertext should be the same as that of the language of the plaintext. Thus, referring to Figure 2.5, there should be one cipher letter with a relative frequency of occurrence of about 12.7%, one with about 9.06%, and so on. If only a single message is available for analysis, we would not expect an exact match of this small sample with the statistical profile of the plaintext language. Nevertheless, if the correspondence is close, we can assume a monoalphabetic substitution. If, on the other hand, a Vigenère cipher is suspected, then progress depends on determining the length of the keyword, as will be seen in a moment. For now, let us concentrate on how the keyword length can be determined. The important insight that leads to a solution is the following: If two identical sequences of plaintext letters occur at a distance that is an integer multiple of the keyword length, they will generate identical ciphertext sequences. In the foregoing example, two instances of the sequence "red" are separated by nine character positions. Consequently, in both cases, r is encrypted using key letter e, e is encrypted using key letter p, and d is encrypted using keyM letter t. Thus, in both cases the ciphertext sequence is VTW. An analyst looking at only the ciphertext would detect the repeated sequences VTW at a displacement of 9 and make the assumption that the keyword is either three or nine letters in length. The appearance of VTW twice could be by chance and not reflect identical plaintext letters encrypted with identical key letters. However, if the message is long enough, there will be a number of such repeated cipher text sequences. By looking for common factors in the displacements of the various sequences, the analyst should be able to make a good guess of the keyword length. Solution of the cipher now depends on an important insight. If the keyword length is N, then the cipher, in effect, consists of N monoalphabetic substitution ciphers. For example, with the keyword DECEPTIVE, the letters in positions 1, 10, 19, and so on are all encrypted with the same monoalphabetic cipher. Thus, we can use the known frequency characteristics of the plaintext language to attack each of the monoalphabetic ciphers separately. The periodic nature of the keyword can be eliminated by using a nonrepeating keyword that is as long as the message itself. Vigenère proposed what is referred to as an **autokey system**, in which a keyword is concatenated with the plaintext itself to provide a running key. For our example,

key: deceptivewearediscoveredsav
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGKZEIIGASXSTSLVVWLA

The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an AT&T engineer named Gilbert Vernam in 1918. His system works on binary data rather than letters.
The system can be expressed succinctly as follows:

ci = pi + ki where
pi = ith binary digit of plaintext ki =
ith binary digit of key ci = ith binary
digit of cipher text
'+'= exclusive-or (XOR) operation

Thus, the ciphertext is generated by performing the bitwise XOR of the plaintext and the key.

Because of the properties of the XOR, decryption simply involves the same bitwise operation: pi = ci +ki
The essence of this technique is the means of construction of the key. Vernam proposed the use of a running loop of tape that eventually repeated the key, so that in fact the system worked with a very long but repeating keyword. Although such a scheme, with a long key, presents formidable cryptanalytic difficulties, it can be broken with sufficient ciphertext, the use of known or probable plaintext sequences, or both.

One-Time Pad

An Army Signal Corp officer, Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad**, is unbreakable. It produces random output that bears no statistical relationship to the plaintext. Because the ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code. An example should illustrate our point. Suppose that we are using a Vigenère scheme with 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message. Thus, the tableau of Table 2.3 must be expanded to 27 x 27. Consider the ciphertext
ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

We now show two different decryptions using two different keys:

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

Key: pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih

Plaintext: mr mustard with the candlestick in the hall

Cipher text: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

Key: mfugpmiydgaxgoufhklllmhsqdqogtewbqfgyovuhwt

Plaintext: miss scarlet with the knife in the library

Suppose that a cryptanalyst had managed to find these two keys. Two plausible plaintexts are produced. How is the cryptanalyst to decide which is the correct decryption (i.e., which is the correct key)? If the actual key were produced in a truly random fashion, then them cryptanalyst cannot say that one of these two keys is more likely than the other. Thus, there is no way to decide which key is correct and therefore which plaintext is correct.

In fact, given any plaintext of equal length to the ciphertext, there is a key that produces that plaintext. Therefore, if you did an exhaustive search of all possible keys, you would end up with many legible plaintexts, with no way of knowing which was the intended plaintext.

Therefore, the code is unbreakable. The security of the one-time pad is entirely due to the randomness of the key. If the stream of characters that constitute the key is truly random, then the stream of characters that constitute the ciphertext will be truly random. Thus, there are no patterns or regularities that a cryptanalyst can use to attack the ciphertext. In theory, we need look no further for a cipher. The one-time pad offers complete security but, in practice, has two fundamental difficulties:

1. There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.

2. Even more daunting is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

Because of these difficulties, the one-time pad is of limited utility, and is useful primarily for low-bandwidth channels requiring very high security.

1.9 Transposition Techniques

All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher. The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message "meet me after the toga party" with a rail fence of depth 2, we write the following:

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

e t e o a a t

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example, Key: 4 3 1 2 5 6 7 Plaintext: a t t a c k p

o s t p o n e d u n t i l t
w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the ciphertext in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is reencrypted using the same algorithm,

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

To visualize the result of this double transposition, designate the letters in the original plaintext message by the numbers designating their position. Thus, with 28 letters in the message, the original sequence of letters is

01 02 03 04 05 06 07 08 09 10 11 12 13 14

15 16 17 18 19 20 21 22 23 24 25 26 27 28

After the first transposition we have

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

which has a somewhat regular structure. But after the second transposition, we have

17 09 05 27 24 16 12 07 10 02 22 20 03 25

15 13 04 23 19 14 11 01 26 21 18 08 06 28

This is a much less structured permutation and is much more difficult to cryptanalyze.

1.10 Rotor Machines

The example just given suggests that multiple stages of encryption can produce an algorithm that is significantly more difficult to cryptanalyze. This is as true of substitution ciphers as it is of transposition ciphers. Before the introduction of DES, the most important application of the principle of multiple stages of encryption was a class of systems known as rotor machines. The basic principle of the rotor machine is illustrated in Figure 2.7. The machine consists of a set of independently rotating cylinders through which electrical pulses can flow. Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin. For simplicity, only three of the internal connections in each cylinder are shown.

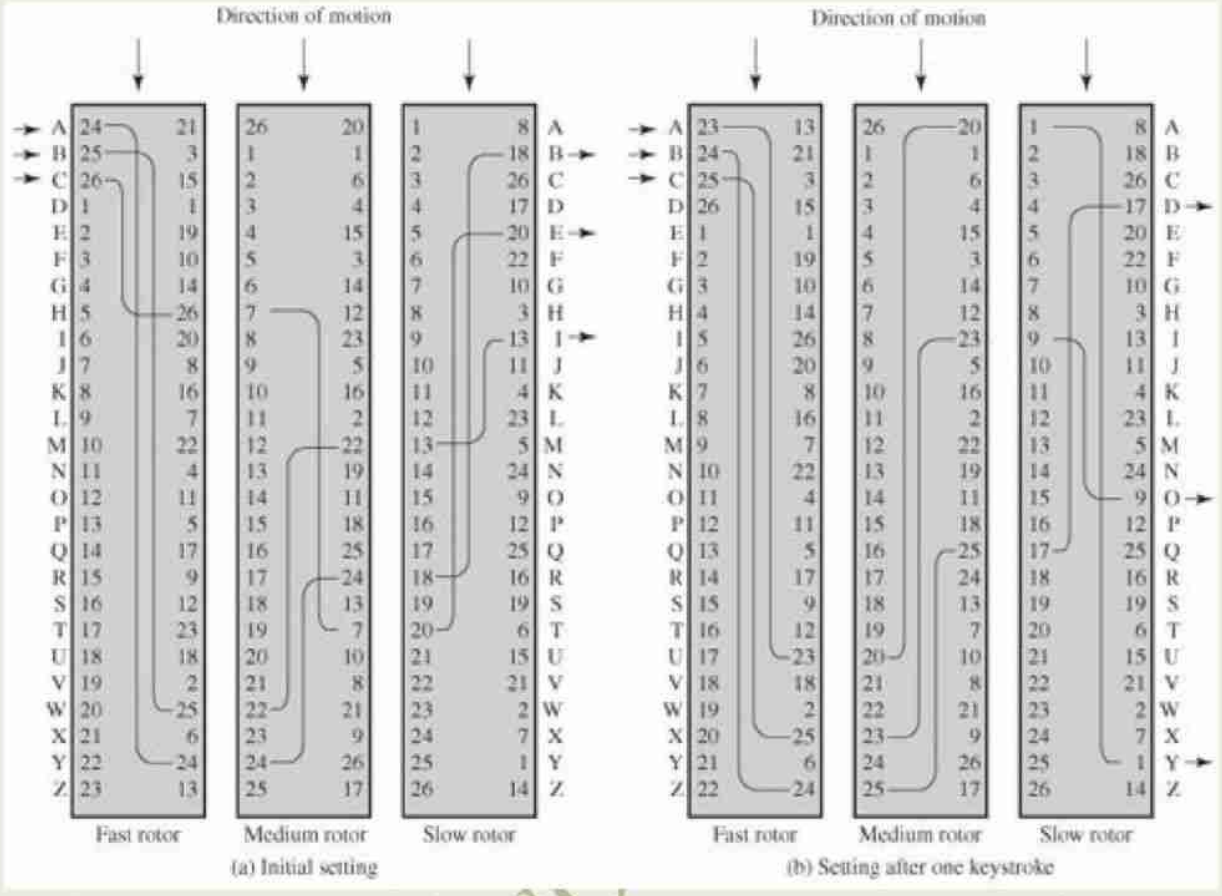


Figure Three-Rotor Machine with Wiring Represented by Numbered Contacts If we associate each input and output pin with a letter of the alphabet, then a single cylinder defines a monoalphabetic substitution. For example, in Figure if an operator depresses the key for the letter A, an electric signal is applied to the first pin of the first cylinder and flows through the internal connection to the twenty-fifth output pin. Consider a machine with a single cylinder. After each input key is depressed, the cylinder rotates one position, so that the internal connections are shifted accordingly. Thus, a different monoalphabetic substitution cipher is defined. After 26 letters of plaintext, the cylinder would be back to the initial position. Thus, we have a polyalphabetic substitution algorithm with a period of 26. A single-cylinder system is trivial and does not present a formidable cryptanalytic task. The power of the rotor machine is in the use of multiple cylinders, in which the output pins of one cylinder are connected to the input pins of the next. Figure 2.7 shows a three-cylinder system. The left half of the figure shows a position in which the input from the operator to the first pin (plaintext letter a) is routed through the three cylinders to appear at the output of the second pin (ciphertext letter B). With multiple cylinders, the one closest to the operator input rotates one pin position with each keystroke. The right half of Figure 2.7 shows the system's configuration after a single keystroke. For every complete rotation of the inner cylinder, the middle cylinder rotates one pin position. Finally, for every complete rotation of the middle cylinder, the outer cylinder rotates one pin position. This is the same type of operation seen with an odometer. The result is that there are 26 x 26 x 26 = 17,576 different

substitution alphabets used before the system repeats. The addition of fourth and fifth rotors results in periods of 456,976 and 11,881,376 letters, respectively.

1.10 Steganography

We conclude with a discussion of a technique that is, strictly speaking, not encryption, namely, steganography. A plaintext message may be hidden in one of two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

Steganography was an obsolete word that was revived by David Kahn. A simple form of steganography, but one that is time-consuming to construct, is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message. For example, the sequence of first letters of each word of the overall message spells out the hidden message.

Various other techniques have been used historically; some examples are the following

Character marking: Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.

Invisible ink: A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

Pin punctures: Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.

Typewriter correction ribbon: Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

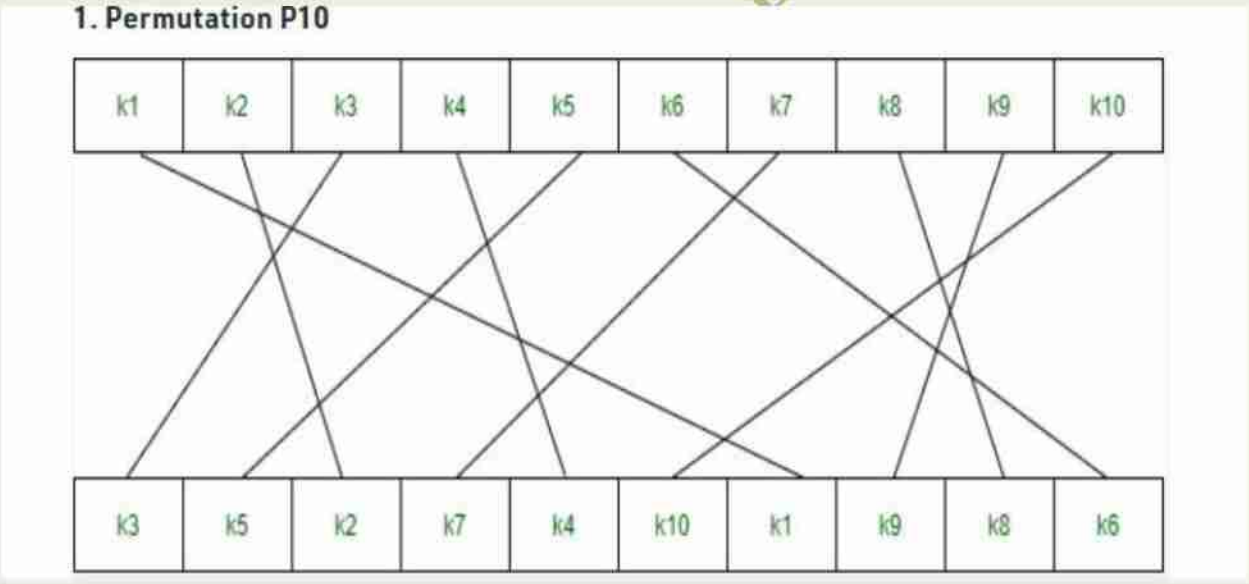
Although these techniques may seem archaic, they have contemporary equivalents. [WAYN93] proposes hiding a message by using the least significant bits of frames on a CD. For example, the Kodak Photo CD format's maximum resolution is 2048 by 3072 pixels, with each pixel containing 24 bits of RGB color information. The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image. The result is that you can hide a 2.3megabyte message in a single digital snapshot. There are now a number of software packages available that take this type of approach to steganography. Steganography has a number of drawbacks when compared to encryption. It requires a lot of overhead to hide a relatively few bits of information, although using some scheme like that proposed in the preceding paragraph may make it more effective. Also, once the system is discovered, it becomes virtually worthless. This problem, too, can be overcome if the insertion method depends on some sort of key. Alternatively, a message can be first encrypted and then hidden using steganography. The advantage of steganography is that it can be employed by parties who have something to lose should the fact of their secret communication (not necessarily the content) be discovered. Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide.

UNIT 2 BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD (DES) 9 Hrs.
Simplified DES- Block Cipher principles – The Data Encryption Standard – The strength of DES – Confidentiality using symmetric encryption – Placement of encryption - Traffic confidentiality – Key distribution - Random number generation.

2.1 SIMPLIFIED DES

Simplified Data Encryption Standard (S-DES) is a simple version of the DES Algorithm. It is similar to the DES algorithm but is a smaller algorithm and has fewer parameters than DES. It was made for educational purposes so that understanding DES would become simpler.
It is a block cipher that takes a block of plain text and converts it into cipher text.
It takes a block of 8 bit. It is a symmetric key cipher i.e. they use the same key for both encryption and decryption. In this article, we are going to demonstrate key generation for s-des encryption and decryption algorithm. We take a random 10-bit key and produce two 8-bit keys which will be used for encryption and decryption.

Key Generation Concept: In the key generation algorithm, we accept the 10-bit key and convert it into two 8 bit keys. This key is shared between both sender and receiver as referred in Fig 2.1.1



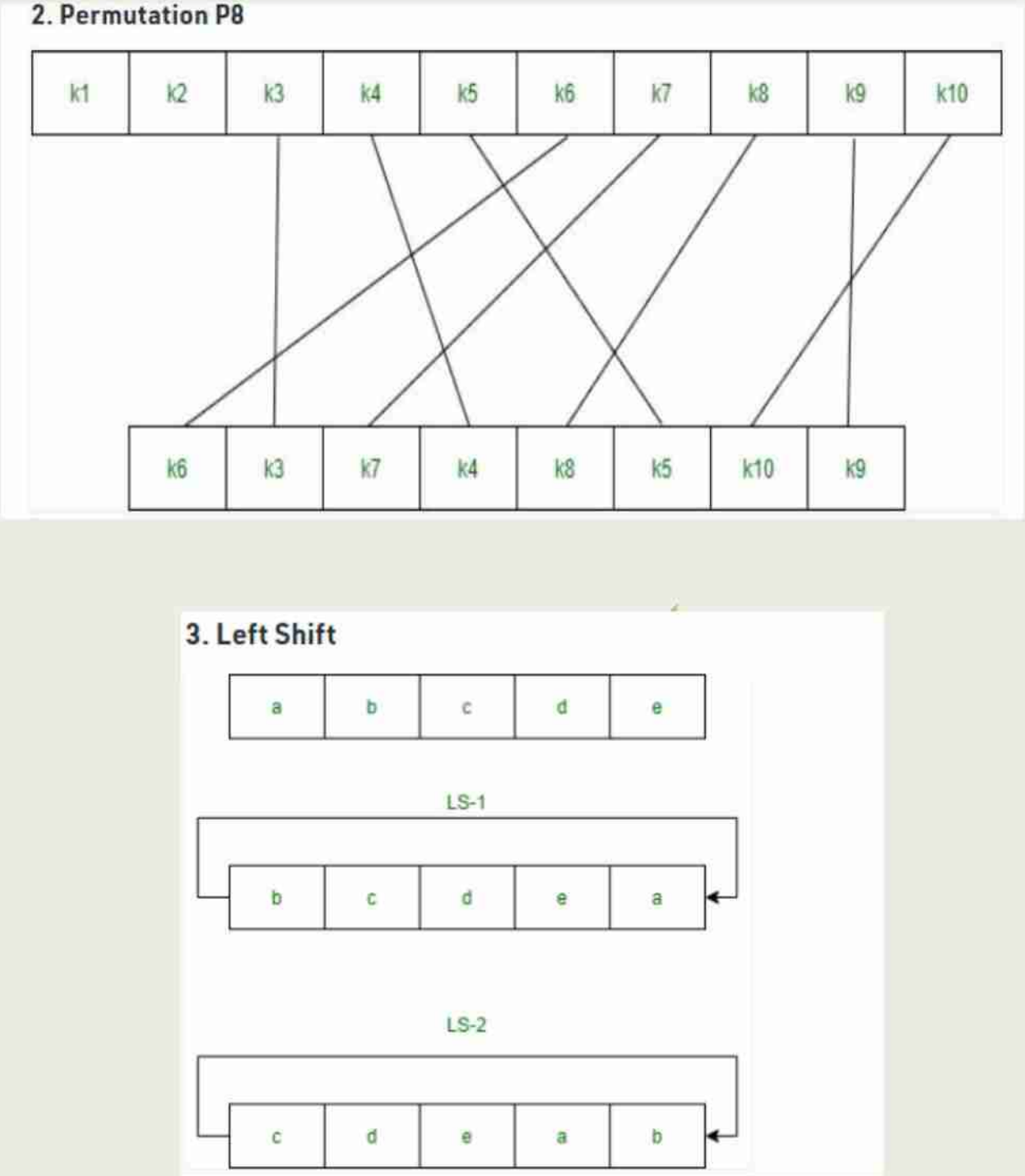


Fig 2.1.1 Key Generation Concept

Step 1: We accepted a 10-bit key and permuted the bits by putting them in the P10 table.
Key = 1 0 1 0 0 0 0 0 1 0

(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (1, 0, 1, 0, 0, 0, 0, 0, 1, 0)

P10 Permutation is: P10(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k3, k5, k2, k7, k4, k10, k1, k9, k8, k6)

After P10, we get 1 0 0 0 0 0 1 1 0 0

Step 2: We divide the key into 2 halves of 5-bit each.
l=1 0 0 0 0, r=0 1 1 0 0

Step 3: Now we apply one bit left-shift on each key.
l = 0 0 0 0 1, r = 1 1 0 0 0

Step 4: Combine both keys after step 3 and permute the bits by putting them in the P8 table. The output of the given table is the first key K1.
After LS-1 combined, we get 0 0 0 0 1 1 1 0 0 0

P8 permutation is: P8(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k6, k3, k7, k4, k8, k5, k10, k9)

After P8, we get Key-1 : 1 0 1 0 0 1 0 0

Step 5: The output obtained from step 3 i.e. 2 halves after one bit left shift should again undergo the process of two-bit left shift.
Step 3 output - l = 0 0 0 0 1, r = 1 1 0 0 0

After two bit shift - l = 0 0 1 0 0, r = 0 0 0 1 1

Step 6: Combine the 2 halves obtained from step 5 and permute them by putting them in the P8 table. The output of the given table is the second key K2.
After LS-2 combined = 0 0 1 0 0 0 0 0 1 1

P8 permutation is: P8(k1, k2, k3, k4, k5, k6, k7, k8, k9, k10) = (k6, k3, k7, k4, k8, k5, k10, k9)

After P8, we get Key-2 : 0 1 0 0 0 0 1 1

Final Output:
Key-1 is: 1 0 1 0 0 1 0 0

Key-2 is: 0 1 0 0 0 0 1 1

The process of encrypting a plan text into an encrypted message with the use of S-DES has been divided into multi-steps which may help you to understand it as easily as possible.

Points should be remembered.

1. It is a block cipher.
2. It has 8-bits block size of plain text or cipher text.
3. It uses 10-bits key size for encryption.
4. It is a symmetric cipher.
5. It has Two Rounds.

Let's start the game!

Key Generation of S-DES or How to Generate the Key of Simplified DES

First and foremost, we need to generate a key. With the help of this key we will encrypt the message. Now the interesting question is, how to generate the key, and where the key is to be used. Just follow the steps.

Step 1:

Just select a random key of 10-bits, which only should be shared between both parties which means sender and receiver.
As I selected below!

Select key:1010000010

Note:You can select any random number of 10-bits.

Step 2:

Put this key into P.10 Table and permute the bits.

P.10 **Table:**

Input	1	2	3	4	5	6	7	8	9	10
Output Should be	3	5	2	7	4	10	1	9	8	6

As I put key into P.10 Table.

Input	1	0	1	0	0	0	0	0	1	0
Output	1	0	0	0	0	0	1	1	0	0

Now the output will be:

Key: 1000001100

Step 3:

Divide the key into two halves, left half and right half;
{1 0 0 0 0} | {0 1 1 0 0}

Step 4:

Now apply the one bit Round shift on each half:
Before round shift: {10000} | {01100}
After round shift: {00001} | {11000}
The output will be:
{0 0 0 0 1} {1 1 0 0 0}

Step 5:

Now once again combine both halve of the bits, right and left. Put them into the P8 table. What you get, that will be the K1 or First key.
Combine: 0 0 0 0 1 1 1 0 0 0

Permute into 8bit table:

P8-Table

Input	1	2	3	4	5	6	7	8	9	10
Combine-bits	0	0	0	0	1	1	1	0	0	0
Output Should be	6	3	7	4	8	5	10	9		
Output bits	1	0	1	0	0	1	0	0		

See the table the 1 and 2 number of bits are removed and other are permuted, as 6 in place of one, 9 in place of 8 and so on.

The output and K1 or key One will be:

K1=1 0 1 0 0 1 0 0

Step6:

As we know S-DES has two round and for that **we also need two keys**, one key we generate in the above steps (step 1 to step 5). Now we need to **generate a second** bit and after that we will move to encrypt the plain text or message.

It is simple to generate the second key. Simply, go in **step 4** copy both halves, each one consists of 5 bits. But be careful on the taking of bits. Select those halves which are output of first round shift, don't take the bits which are not used in the first round. In simple words, take the output of first round shift in above **step 4.**

Which are: {00001} | {11000}

Step 7:

Now just apply two round shift circulate on each half of the bits, which means to change the position of two bits of each halves.
left half: 00001
Right half: 11000

After the two rounds shift on each half out-put of each half will be.
Left half: 00100
Right half: 00011
Combine both together: As: 0 0 1 0 0 – 0 0 0 1 1

Step 8:

Now put the bits into 8-P Table, what you get, that will be your second key. Table is also given in **step 5**.

But here the combinations of bits are changed because of two left round shift from step 5. Check it in depth.
Combine bits: 0 0 1 0 0 0 0 0 1 1

P.8 **Table**

Input	1	2	3	4	5	6	7	8	9	10
Combine-bits	0	0	1	0	0	0	0	0	1	1
Output Should be	6	3	7	4	8	5	10	9		
Output bits	0	1	0	0	0	0	1	1		

The output of the bits are your Second key or K2:
K2: 0 1 0 0 0 0 1 1

Finally we create both keys successfully:

K1: 1 0 1 0 0 1 0 0 (see in step 5)
K2: 0 1 0 0 0 0 1 1 (see in step 8)

How To Encrypt the Plain Text into Cipher Text in S-DES After Generating Keys.

Now, let's start Encryption of plain text into cipher text.

Encryption of Plain text into Cipher text in S-DES:

Come on do it, **step by step**.
Note: the size of input text is 8 bit and output also will be 8-bit. Or the block size is 8-bit/one byte always.

Step 1:

Suppose this is our plain text in binary which is 8-bit.

Plain text: 01110010

Step 2:

Put the plain text into IP-8(initial permutation) table and permute the bits.

IP-8 **table**

Bits number	1	2	3	4	5	6	7	8
Bits to be permuted	0	1	1	1	0	0	1	0
Permute the bits	2	6	3	1	4	8	5	7
Permuted bits	1	0	1	0	1	0	0	1

Output is: 1 0 1 0 1 0 0 1

Step 3:

Now break the bits into two halves, each half will consist of 4 bits. The halves will be right and left.
Two Halves of the bits:

Left half {1 0 1 0} -right half {1 0 0 1}

Step 4:

Take the right 4 bits and put them into E.P (expand and per-mutate) Table.
Bits of right half: 1001

E.P **Table**

Right half bits	1	0	0	0				
Number	1	2	3	4	5	6	7	8
Expand bits	4	1	2	3	2	3	4	1
Output of bits	0	1	0	0	0	0	0	1

Output of right four bits will be 8 bits, after their expanding with the help of E.P table.

O-P: 0 1 0 0 0 0 0 1

Step 5: Now, just take the output and XOR it with First key Or K 1 (which we created in previous topic that is how to generate key.).

XOR (⊕) them:

O-p: 0 1 0 0 0 0 0 1
⊕
K1: 1 0 1 0 0 1 0 0
Output of XOR: **1 1 1 0 0101**

Step 6:

Once again split the output of XOR's bit into two halves and each half will consist of 4 bits.

Splitting them into two halves:

Left half :{ 1 1 1 0} right half :{ 0101}
Now put the each half into the **s-boxes**, there is only two s-boxes. **S-0 and S-1.**

S-0

Col	0	1	2	3
Rows				
0	01	00	11	10
1	11	10	01	00
2	00	10	01	11
3	11	01	11	10

S-1

Col	0	1	2	3
Rows				
0	00	01	10	11
1	10	00	01	11
2	11	00	01	00
3	10	01	00	11

Note: put the left half into S-0 box and put the right half into S-1 Box.

But how to put them in into S-Boxes?

Take any half, (but don't forget the above mentioned note..).
The most first and most last bit will be consider the row and other remaining, which are, 2 and 3, will be considered the columns.
See, here I'm taking the left half: which is 1 1 1 0.

Now I will take First and last bit which are: 1 and 0. These will be row.

And I also will take 2nd and 3rd bits which are: 11. These will be column number.
10 means=2rd **row**
11 means = 3rd **col**

See below how it will be the second row, and 3rd column. Please, remember the IP Addressing, such as 2^8 for 255.
120^1 =2+0=2
$1^2 1^1$ =2+1=3

Let's check the 2nd row and 3rd column.

For **left half** we check the in S**-0** . In which 2nd row and 3rd column.

The output is 00 for left half;

Now let's take the **right half** and find the output of the right of in **S-1 box**.
Right half: 0101
Row: 0 1= $0^2 1^1$=0+1=1
Col: 1 0=$1^2 0^1$=2+1=3

Which means the value will be in 1st row and 3rd column, let's in **check S-1.**
The output will be: 11 for left half.

Step 7:

Now combine these two halves together.

Left half: {00} and right half: {11}
It will be: 0 0 1 1

Step 8: Now take these 4 bits and put them in **P-4** (permutation 4) table and get the result.

P **4** **Table**

Numbers	1	2	3	4
Input	0	0	1	1
Output should be	2	4	3	1
Out-Put	0	1	1	0

Now the output is: 0 1 1 0

Step 9:

Now get XOR the output with left 4 bits of Initial Per-mutation. The left bits of initial per-mutation are in **step 3**, which are **1 0 1 0**.(please, in step 3).
Let them to be XOR.

0 1 1 0
⊕
1 0 1 0

Out-put will be: 1 1 0 0

Step 10:

Now get the right half of the initial permutation, which is **step 3,** and combine that with this out- put.

The out-put of XOR in **step 9:** 1 1 0 0
Right half of IP (initial permutation): 1 0 0 1
Let's combine both. 1 1 0 0 – 1 0 0 1= 1 1 0 0 1 0 0 1
Now the output is 8 bits.: **1 1 0 0 1 0 0 1**

Step 11: Now once again break the out-put into two halves, left and right;

Left: {1 1 0 0} right: {1 0 0 1}

Step 12:

Now swap both halves, which means put the left half in place of right and vice versa.

Result:
Left half: {1 0 0 1} right half: {1 1 0 0}

Step 13:

Now let's take these halves and once again start the same procedure from **step 2** or initial Permutation, **BUT** be careful on using key in this stage we use second key or K2 (not K1). And put that into IP^{-1} (IP inverse) Table. What you get will be your final cipher text.

Let me to do it in brief. You should check carefully what I did.
Ø 1 0 0 1 11 0 0

After initial permutation according to **IP-8 table (see step 2)**
Out-put will be: 0 1 0 1 1 0 1 0

Ø Now break it into two halves
Left {0 1 0 1} right {1 0 1 0}

Ø Now Take the right 4bits and put them into **EP table**, and get the result of 8 bits.
It will be: 0 1 0 1 0 1 0 1

Ø Let **XOR it with K2.**
K2: 0 1 0 0 0 0 1 1
⊕

Out-put of **EP**: 0 1 0 1 0 1 0 1
Out- Put: **0 0 0 1 0 1 1 0**

Ø Once again split the output of XOR's bit into two halves:
Left: {0 0 0 1} right: {0 1 1 0}

Ø Now put each half in S-Boxes, which are S-0 and S-1:

Note: put the left half into S-0 box and put the right half into S-1 Box.

Before that get Rows and column:

Left: 0 0 0 1
Row: 0 1= 1
Col: 0 0 = 0

Let find the row and column of left, in S-0, the value is row number one and col number Zero.

It will be. 1 1
Ø now check the right half: 0 1 1 0
Row: 0 0=0
Col: 1 1= 3

Let's find the row and column of right, in S-1, the value is row number zero and col number three.

It will be: 11
Ø Now combine both halves together.
It will be: 1 1 1 1

Ø Now take these 4 bits and put them in **P-4** (**Per-mutation** 4) table and get the result.

The out-put also will be: 1111 after permutation.

Ø Now get it XOR with left 4 bits of Initial Per-Mutation.
1 1 1 1
XOR
0 1 0 1

Out-put:1 0 1 0

Ø Now get the right half of the initial permutation and combine that with this out- put.
1 0 1 0 - 0 1 1 0

Ø Now once again break the it into two halves, left and right
Left: {1 0 1 0} right: {0 1 1 0}

Ø Now **swap both halves,** which means put the left half in place of right and vice versa.
0 1 1 0 1 0 1 0

Ø Now put it into IP^{-1} Table which is :

IP^{-1} Table

Numbers	1	2	3	4	5	6	7	8
input	0	1	1	0	1	0	1	0
Out-put to be	2	6	3	1	4	8	5	7
Out-Put	1	0	1	0	0	0	1	1

Out-Put is the cipher text. Which is: 1 0 1 0 0 0 1 1

Finally we Encrypted successfully our **plain text: 01110010** into **cipher text** which is:

1 0 1 0 0 0 1 1

2.2 BLOCK CIPHER PRINCIPLES

Block ciphers are built in the Feistel cipher structure. Block cipher has a specific number of rounds and keys for generating cipher text. For defining the complexity level of an algorithm few design principles are to be considered.
These are explained as following below:

1. **Number of Rounds** – The number of Rounds is regularly considered in design criteria, it just reflects the number of rounds to be suitable for an algorithm to make it more complex, in DES we have 16 rounds ensuring it to be more secure while in AES we have 10 rounds which make it more secure.
2. **Design of function F** – The core part of the Feistel Block cipher structure is the Round Function. The complexity of cryptanalysis can be derived from the Round function i.e. the increasing level of complexity for the round function would be greatly contributing to an increase in complexity.

To increase the complexity of the round function, the avalanche effect is also included in the round function, as the change of a single bit in plain text would produce a mischievous output due to the presence of avalanche effect.

3. **Key schedule algorithm** – In Feistel Block cipher structure, each round would generate a sub-key for increasing the complexity of cryptanalysis. The Avalanche effect makes it more complex in deriving sub-key. Decryption must be done very carefully to get the actual output as the avalanche effect is present in it.

Block Cipher Modes of Operation

There are five important block cipher modes of operation defined by NIST. These five modes of operation enhance the algorithm so that it can be adapted by a wide range of applications which uses block cipher for encryption.

1. Electronic Code Book Mode
2. Cipher Block Chaining Mode
3. Cipher Feedback Mode
4. Output Feedback Mode
5. Counter Mode

1. Electronic Feedback Mode

This is considered to be the easiest block cipher mode of operation. In electronic codebook mode (ECB) the plain text is divided into the blocks, each of 64-bit. Each block is encrypted one at a time to produce the cipher block. The same key is used to encrypt each block referred in Fig 2.2.1

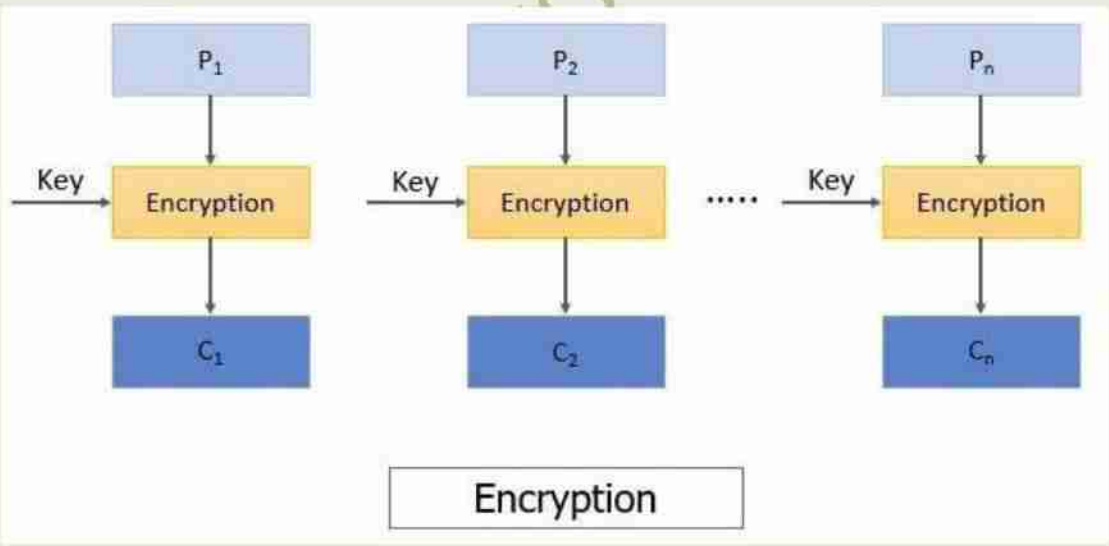


Fig 2.2.1 Block Cipher

When the receiver receives the message i.e. cipher text. This cipher text is again divided into blocks, each of 64-bit and each block is decrypted independently one at a time to obtain the corresponding plain text block. Here also the same key is used to decrypt each block which was used to encrypt each block.

As the same key used to encrypt each block of plain text there arises an issue that for a repeating plain text block it would generate the same cipher and will ease the cryptanalysis to crack the algorithm. Hence, ECB is considered for encrypting the small messages which have a rare possibility of repeating text.

2. Cipher Block Chaining Mode

To overcome the limitation of ECB i.e. the repeating block in plain text produces the same ciphertext, a new technique was required which is Cipher Block Chaining (CBC) Mode. CBC confirms that even if the plain text has repeating blocks its encryption won't produce same cipher block.

To achieve totally different cipher blocks for two same plain text blocks **chaining** has been added to the block cipher. For this, the result obtained from the encryption of the first plain text block is fed to the encryption of the next plaintext box.

In this way, each cipher text block obtained is dependent on its corresponding current plain text block input and all the previous plain text blocks. But during the encryption of first plain text block, no previous plain text block is available so a random block of text is generated called **Initialization vector**.

Now let's discuss the encryption steps of CBC

Step 1: The initialization vector and first plain text block are XORed and the result of XOR is then encrypted using the key to obtain the first cipher text block.

Step 2: The first cipher text block is fed to the encryption of the second plain text block. For the encryption of second plain text block, first cipher text block and second plain text block is XOR and the result of XOR is encrypted using the same key in step 1 to obtain the second cipher text block.

Similarly, the result of encryption of second plain text block i.e. the second cipher text block is fed to the encryption of third plain text block to obtain third cipher text block. And the process continues to obtain all the cipher text blocks.

You can see the steps of CBC in the figure 2.2.2 below:

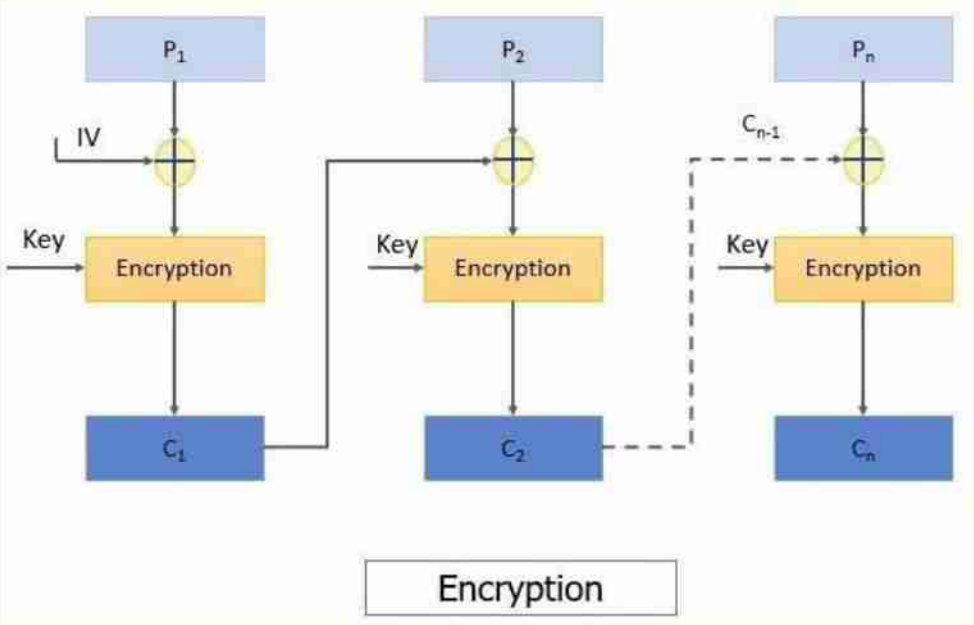


Fig 2.2.2 CBC

Decryption steps of CBC:

Step 1: The first ciphertext block is decrypted using the same key that was used for encrypting all plain text blocks. The result of decryption is then XORed with the initialization vector (IV) to obtain the first plain text block.

Step 2: The second ciphertext block is decrypted and the result of decryption is XORed with the first ciphertext block to obtain the second plain text block. And the process continues till all plain text blocks are retrieved.

There was a limitation in CBC that if we have two identical messages and if we use the same IV for both the identical message it would generate the same ciphertext block.

3. Cipher Feedback Mode

All applications may not be designed to operate on the blocks of data, some may be **character or bit-oriented**. Cipher feedback mode is used to operate on smaller units than blocks.

Step 1: Here also we use initialization vector, IV is kept in the shift register and it is encrypted using the key.

Step 2: The left most s bits of the encrypted IV is then XORed with the first fragment of the plain text of s bits. It produces the first ciphertext C1 of s bits.

Step 3: Now the shift register containing initialization vector performs left shift by s bits and s bits C1 replaces the rightmost s bits of the initialization vector.

Then again, the encryption is performed on IV and the leftmost s bit of encrypted IV is XORed with the second fragment of plain text to obtain s bit ciphertext C2.

4. Output Feedback Mode

The output feedback (OFB) mode is almost similar to the CFB. The difference between CFB and OFB is that unlike CFB, in OFB the encrypted IV is fed to the encryption of next plain text block. The other difference is that CFB operates on a stream of bits whereas OFB operates on the block of bits.

Steps for encryption:

Step 1: The initialization vector is encrypted using the key.

Step 2: The encrypted IV is then XORed with the plain text block to obtain the ciphertext block.

5. Counter Mode

It is similar to OFB but there is no feedback mechanism in counter mode. Nothing is being fed from the previous step to the next step instead it uses a sequence of number which is termed as a **counter** which is input to the encryption function along with the key. After a plain text block is encrypted the counter value increments by 1.

Steps of encryption:

Step1: The counter value is encrypted using a key.

Step 2: The encrypted counter value is XORed with the plain text block to obtain a cipher text block.

To encrypt the next subsequent plain text block the counter value is incremented by 1 and step 1 and 2 are repeated to obtain the corresponding cipher text as referred in Fig 2.2.3

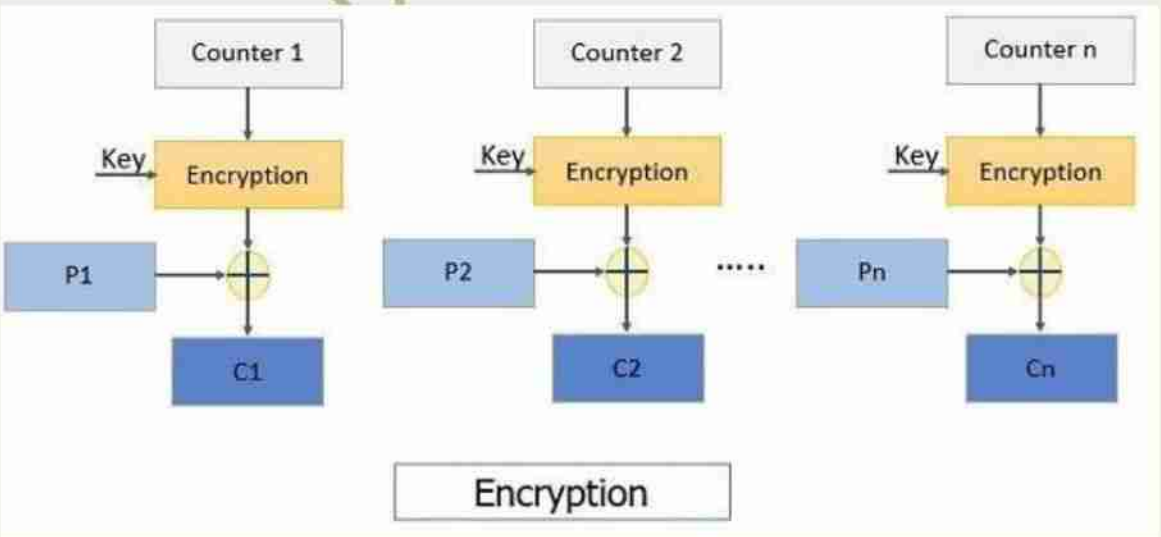


Fig 2.2.3 Counter Mode

2.3 THE DATA ENCRYPTION

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit.

Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following Fig 2.3.1

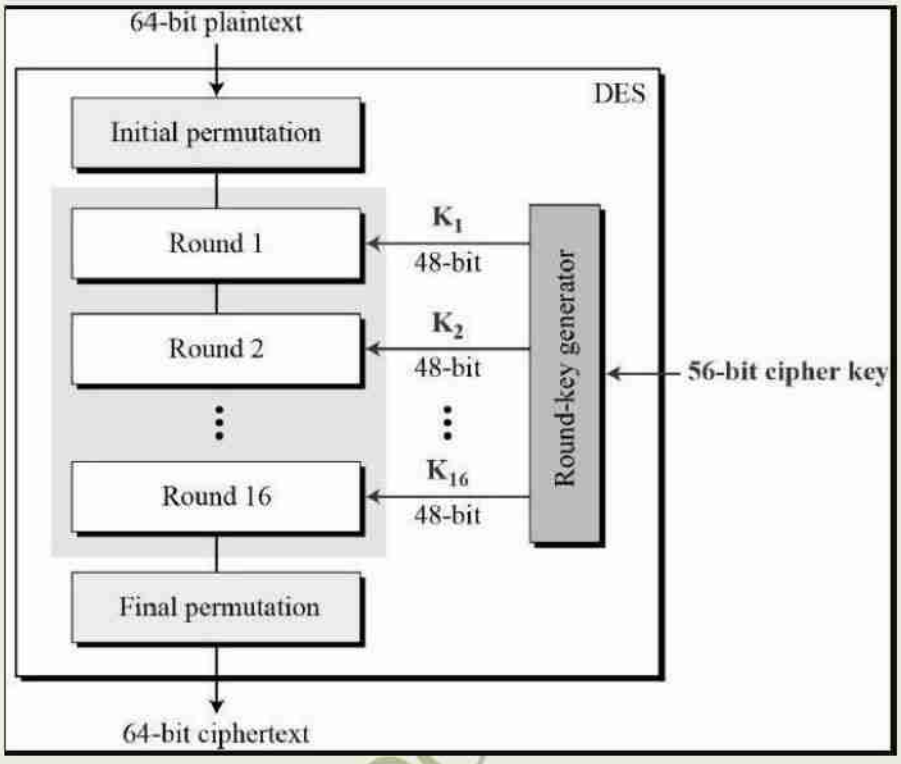


Fig 2.3.1 DES

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

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

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are shown as follows (Fig 2.3.2)

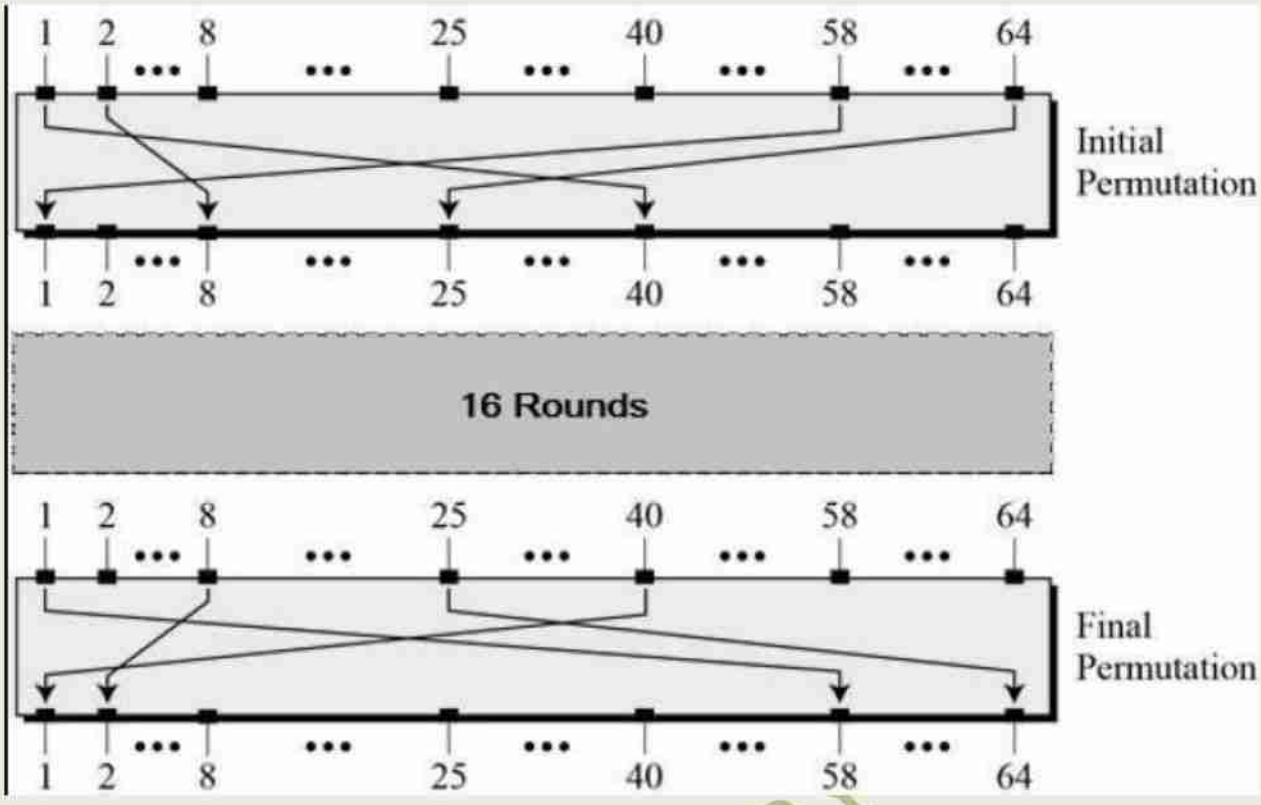


Fig 2.3.2 Initial and Final Permutation

Round Function

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

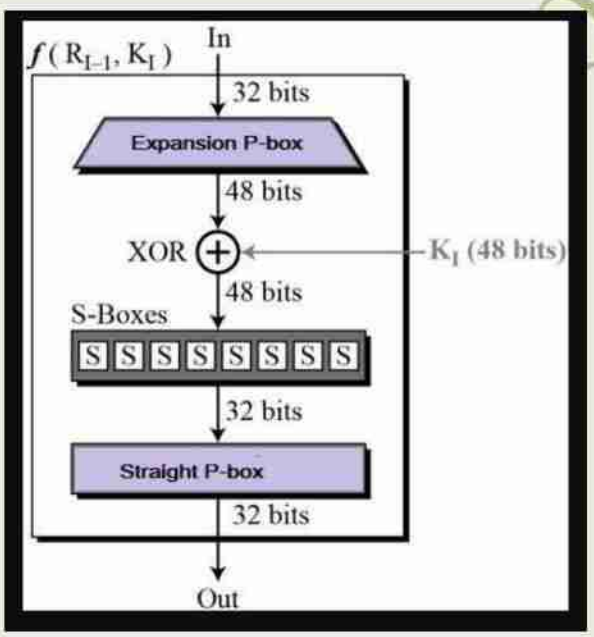


Fig 2.3.3 Round Function

- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the

following illustration − he graphically depicted permutation logic is generally described as table in DES specification illustrated as shown(Fig 2.3.4)

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

Fig 2.3.4 Expansion Permutation Box

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −

DES Analysis

The DES satisfies both the desired properties of block cipher. These two properties make cipher very strong.

- **Avalanche effect** − A small change in plaintext results in the very great change in the cipher text.
- **Completeness** − Each bit of cipher text depends on many bits of plaintext.

During the last few years, cryptanalysis have found some weaknesses in DES when key selected are weak keys. These keys shall be avoided.

DES has proved to be a very well designed block cipher. There have been no significant cryptanalytic attacks on DES other than exhaustive key search.

2.4 THE STRENGTH OF DES

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA).7 For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption. DES Encryption The overall scheme for DES encryption is illustrated in Figure 3.5. As with any encryption scheme, there are two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintext must be 64 bits in length and the key is 56 bits in length. Looking at the left-hand side of the figure,

we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input.

This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the pre output. Finally, the pre output is passed through a permutation [IP1] that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutations(referred in Fig 2.4.1)

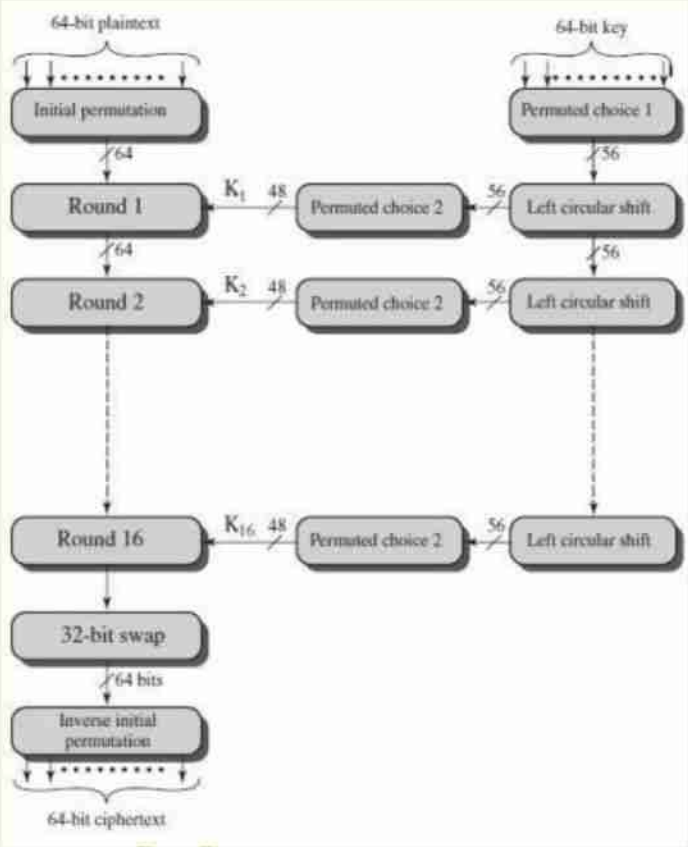
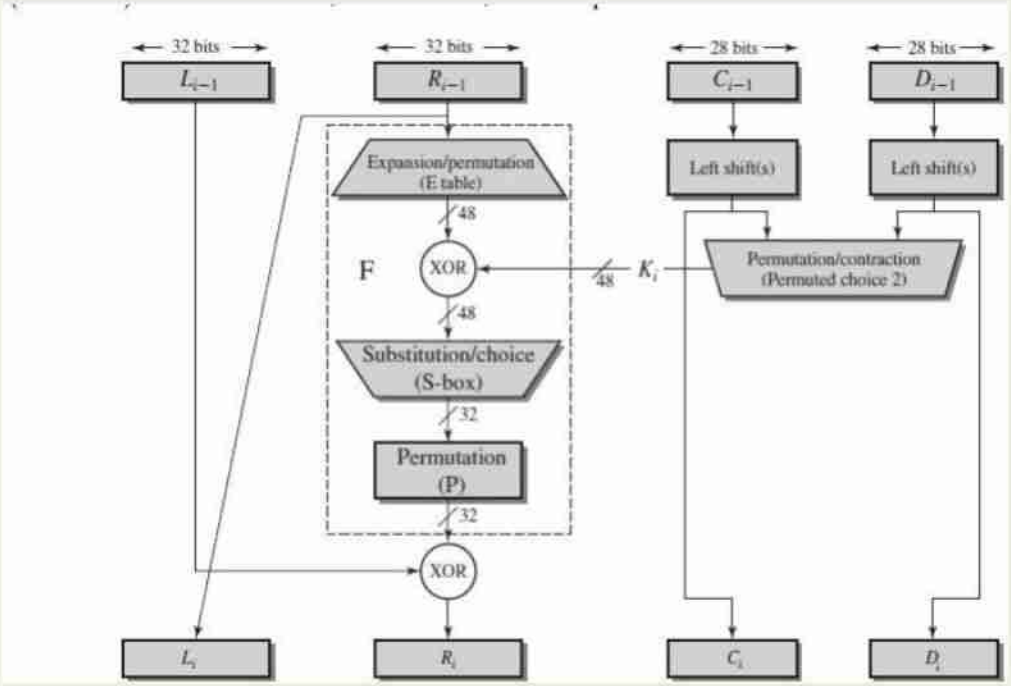
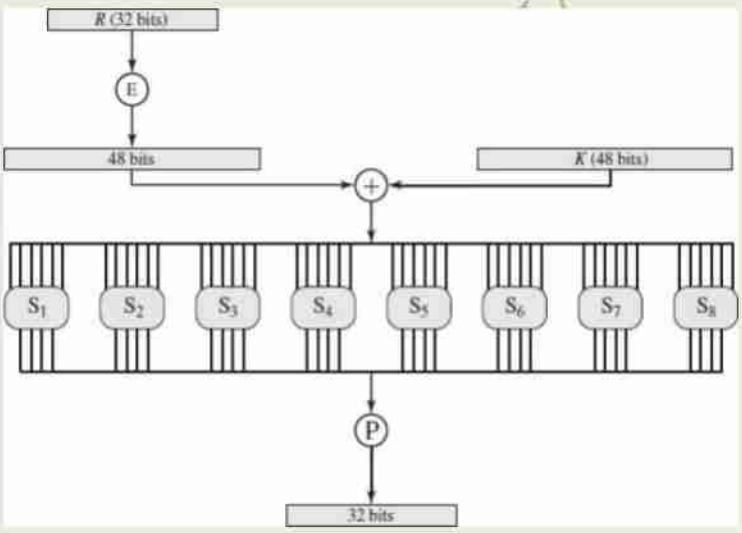


Fig 2.4.1 Strength of DES

The right-hand portion of Figure 3.5 shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a subkey (Ki) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits. INITIAL PERMUTATION The initial permutation and its inverse are defined by tables, as shown in Tables 3.2a and 3.2b, respectively. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered from 1 to 64. The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits. DETAILS OF SINGLE ROUND Figure 3.6 shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labelled L (left) and R (right). Li = Ri-1 Ri = Li-1 $\oplus$ {F(Ri-1, Ki) }


32 bits
32 bits
28 bits
28 bits
L_{i-1}
R_{i-1}
C_{i-1}
D_{i-1}
Expansion/permutation
(E table)
Left shift(s)
Left shift(s)
48
F
XOR
48
K_i
Permutation/contraction
(Permuted choice 2)
48
Substitution/choice
(S-box)
32
Permutation
(P)
32
XOR
L_i
R_i
C_i
D_i


R (32 bits)
E
48 bits
K (48 bits)
+
S1
S2
S3
S4
S5
S6
S7
S8
P
32 bits

(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation (IP^{-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

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

Fig 2.4.2 Single round of DES

2.5 CONFIDENTIALITY USING SYMMETRIC ENCRYPTION

Providing confidentiality through the use of secret-key encryption has historically been the focus of cryptology. This topic remains important in itself, though other considerations have emerged in the last few decades. An understanding of the issues involved here clarifies those in other applications of encryption and helps to motivate the development of public-key encryption.

Issues involved: What should be encrypted? Where should encryption be done? Two approaches: Link encryption End-to-end encryption to make the decisions, one should first examine the potential locations of security attacks.

Consider a user workstation in a typical business organization. The points of vulnerability include: The LAN that the workstation is attached to: eavesdropping on the LAN, which is typically a broadcast network. The Wiring closet: tapping the wires. Communications links out of the Wiring closet: invasive or inductive tapping. Processors along the path to the outside: modifying the

hardware or software,etc.

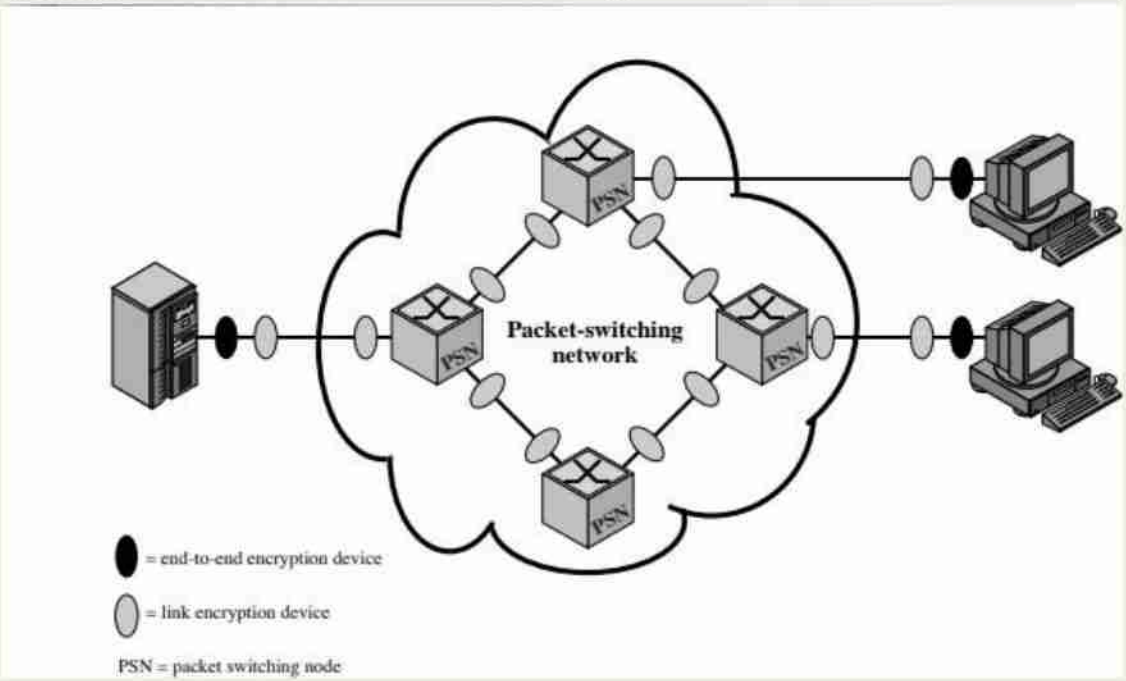


Fig 2.5.1 Encryption in Packet-Switching Networks

Link Encryption Each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. The message must be decrypted each time it enters a packet switch. Thus, the message is vulnerable at each switch. Many keys must be provided. However, each key needs be distributed to only two nodes(Referred in Fig 2.5.1)

End-to-End Encryption The encryption process is carried out at the two end systems. The source and the destination share a key. This plan seems to secure the transmission against attacks on the network links or switches. There is, however, still a weak spot. The source may encrypt only the user data portion, but must leave the header in the clear. With end-to-end encryption, the user data are secure, but the traffic pattern is not. A certain degree of authentication is also provided.

Link vs. End-to-End Encryptions Link Encryption End-to-End Encryption Security within End Systems and Intermediate Systems Message exposed in sending host Message exposed in intermediate nodes Message encrypted in sending host Message encrypted in intermediate nodes Role of User Applied by sending host

Transparent to user Host maintains encryption facility One facility for all users Can be done in hardware All or no messages encrypted Applied by sending process User applies encryption User must determine algorithm

Users selects encryption scheme Software implementation User chooses to encrypt, or not, for each message Implementation Concerns Requires one key per (host-intermediate node) pair and (intermediate node intermediate node) pair Provides host authentication Requires one key per user pair Provides user authentication

Deploying End-to-End Encryption Possible choices: The network layer or the transport layer one key for each pair of end systems cannot cross internetwork boundaries

The application layer many keys needed: one key for each pair of users can cross inter network boundaries

2.6 PLACEMENT OF ENCRYPTION

If encryption is to be used to counter attacks on confidentiality, we need to decide what to encrypt and where the encryption function should be located. To begin, this section examines the potential locations of security attacks and then looks at the two major approaches to encryption placement: link and end to end.

Potential Locations for Confidentiality Attacks

As an example, consider a user workstation in a typical business organization. In Figure 2.6.1 suggests the types of communications facilities that might be employed by such a workstation and therefore gives an indication of the points of vulnerability.

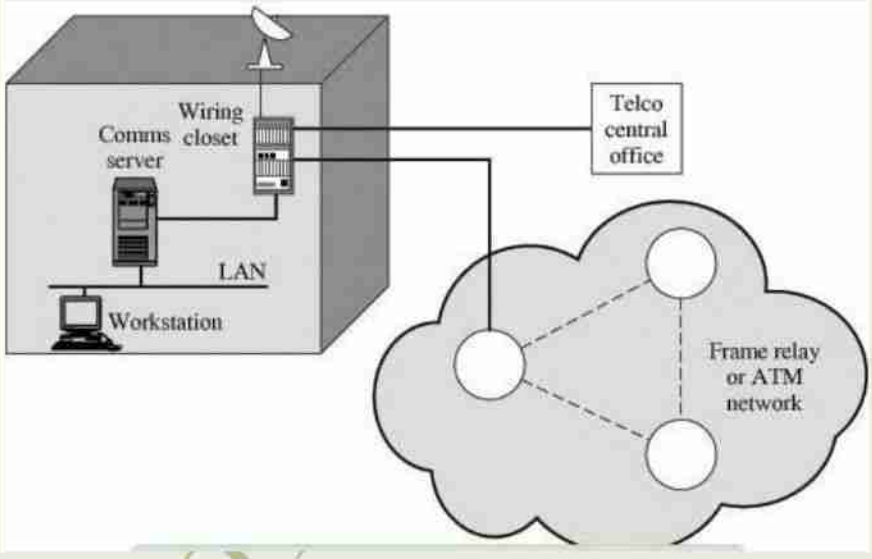


Fig 2.6.1 Point of Vulnerability

In most organizations, workstations are attached to local area networks (LANs). Typically, the user can reach other workstations, hosts, and servers directly on the LAN or on other LANs in the same building that are interconnected with bridges and routers. Here, then, is the first point of vulnerability. In this case, the main concern is eavesdropping by another employee. Typically, a LAN is a broadcast network: Transmission from any station to any other station is visible on the LAN medium to all stations. Data are transmitted in the form of frames, with each frame containing the source and destination address. An eavesdropper can monitor the traffic on the LAN and capture any traffic desired on the basis of source and destination addresses. If part or all of the LAN is wireless, then the potential for eavesdropping is greater.

Furthermore, the eavesdropper need not necessarily be an employee in the building. If the LAN, through a communications server or one of the hosts on the LAN, offers a dial-in capability, then it is possible for an intruder to gain access to the LAN and monitor traffic.

Access to the outside world from the LAN is almost always available in the form of a router that connects to the Internet, a bank of dial-out modems, or some other type of communications server.

From the communications server, there is a line leading to a wiring closet. The wiring closet serves as a patch panel for interconnecting internal data and phone lines and for providing a staging point for external communications.

The wiring closet itself is vulnerable. If an intruder can penetrate to the closet, he or she can tap into each wire to determine which are used for data transmission. After isolating one or more lines, the intruder can attach a low-power radio transmitter. The resulting signals can be picked up from a nearby location (e.g., a parked van or a nearby building).

Several routes out of the wiring closet are possible. A standard configuration provides access to the nearest central office of the local telephone company. Wires in the closet are gathered into a cable, which is usually consolidated with other cables in the basement of the building. From there, a larger cable runs underground to the central office.

In addition, the wiring closet may provide a link to a microwave antenna, either an earth station for a satellite link or a point-to-point terrestrial microwave link. The antenna link can be part of a private network, or it can be a local bypass to hook in to a long-distance carrier.

The wiring closet may also provide a link to a node of a packet-switching network. This link can be a leased line, a direct private line, or a switched connection through a public telecommunications network. Inside the network, data pass through a number of nodes and links between nodes until the data arrive at the node to which the destination end system is connected.

An attack can take place on any of the communications links. For active attacks, the attacker needs to gain physical control of a portion of the link and be able to insert and capture transmissions. For a passive attack, the attacker merely needs to be able to observe transmissions. The communications links involved can be cable (telephone twisted pair, coaxial cable, or optical fiber), microwave links, or satellite channels. Twisted pair and coaxial cable can be attacked using either invasive taps or inductive devices that monitor electromagnetic emanations. Invasive taps allow both active and passive attacks, whereas inductive taps are useful for passive attacks. Neither type of tap is as effective with optical fiber, which is one of the advantages of this medium. The fiber does not generate electromagnetic emanations and hence is not vulnerable to inductive taps. Physically breaking the cable seriously degrades signal quality and is therefore detectable. Microwave and satellite transmissions can be intercepted with little risk to the attacker. This is especially true of satellite transmissions, which cover a broad geographic area. Active attacks on microwave and satellite are also possible, although they are more difficult technically and can be quite expensive.

n addition to the potential vulnerability of the various communications links, the various processors along the path are themselves subject to attack. An attack can take the form of attempts to modify the hardware or software, to gain access to the memory of the processor, or to monitor the electromagnetic emanations. These attacks are less likely than those involving communications links but are nevertheless a source of risk.

Thus, there are a large number of locations at which an attack can occur. Furthermore, for wide area communications, many of these locations are not under the physical control of the end user. Even in the case of local area networks, in which physical security measures are possible, there is always the threat of the disgruntled employee.

Link versus End-to-End Encryption

The most powerful and most common approach to securing the points of vulnerability highlighted in the preceding section is encryption. If encryption is to be used to counter these attacks, then we need to decide what to encrypt and where the encryption gear should be located. As Figure 7.2 indicates, there are two fundamental alternatives: link encryption and end-to-end encryption. With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. Although this recourse requires a lot of encryption devices in a large network, its value is clear. One of its disadvantages is that the message must be decrypted each time it enters a switch (such as a frame relay switch) because the switch must read the address (logical connection number) in the packet header in order to route the frame. Thus, the message is vulnerable at each switch. If working with a public network, the user has no control over the security of the nodes.

Several implications of link encryption should be noted. For this strategy to be effective, all the potential links in a path from source to destination must use link encryption. Each pair of nodes that share a link should share a unique key, with a different key used on each link. Thus, many keys must be provided.

With end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data. The data in encrypted form are then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data. This plan seems to secure the transmission against attacks on the network links or switches. Thus, end-to-end encryption relieves the end user of concerns about the degree of security of networks and links that support the communication. There is, however, still a weak spot.

Consider the following situation. A host connects to a frame relay or ATM network, sets up a logical connection to another host, and is prepared to transfer data to that other host by using end-to-end encryption. Data are transmitted over such a network in the form of packets that consist of a header and some user data. What part of each packet will the host encrypt? Suppose that the host encrypts the entire packet, including the header. This will not work because, remember, only the other host can perform the decryption. The frame relay or ATM switch will receive an encrypted packet and be unable to read the header. Therefore, it will not be able to route the packet. It follows that the host may encrypt only the user data portion of the packet and must leave the header in the clear.

Figure 2.6.2 shows the encryption function of the front-end processor (FEP). On the host side, the FEP accepts packets. The user data portion of the packet is encrypted, while the packet header bypasses the encryption process.[1] The resulting packet is delivered to the network. In the opposite direction, for packets arriving from the network, the user data portion is decrypted and the entire packet is delivered to the host. If the transport layer functionality (e.g., TCP) is implemented in the front end, then the transport-layer header would also be left in the clear and the user data portion of the transport protocol data unit is encrypted. The terms red and black are frequently used. Red data are sensitive or classified data in the clear. Black data are encrypted data.

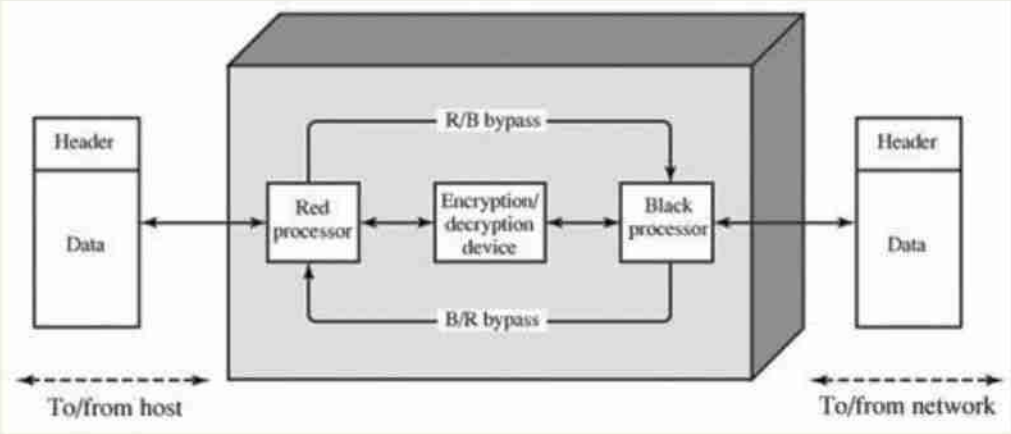


Fig 2.6.2 Front-End Processor Function

2.7 TRAFFIC CONFIDENTIALITY

In some cases, users are concerned about security from traffic analysis. Knowledge about the number and length of messages between nodes may enable an opponent to determine who is talking to whom. This can have obvious implications in a military conflict. Even in commercial applications, traffic analysis may yield information that the traffic generators would like to conceal. [MUFT89] lists the following types of information that can be derived from a traffic analysis attack:

- Identities of partners
- How frequently the partners are communicating
- Message pattern, message length, or quantity of messages that suggest important information is being exchanged
- The events that correlate with special conversations between particular partners

Another concern related to traffic is the use of traffic patterns to create a **covert channel**. A covert channel is a means of communication in a fashion unintended by the designers of the communications facility. Typically, the channel is used to transfer information in a way that violates a security policy. For example, an employee may wish to communicate information to an outsider in a way that is not detected by management and that requires simple eavesdropping on the part of the outsider. The two participants could set up a code in which an apparently legitimate message of a less than a certain length represents binary zero, whereas a longer message represents a binary one. Other such schemes are possible.

Link Encryption Approach

With the use of link encryption, network-layer headers (e.g., frame or cell header) are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system. An effective countermeasure to this attack is traffic padding, referred in Fig 2.7.1

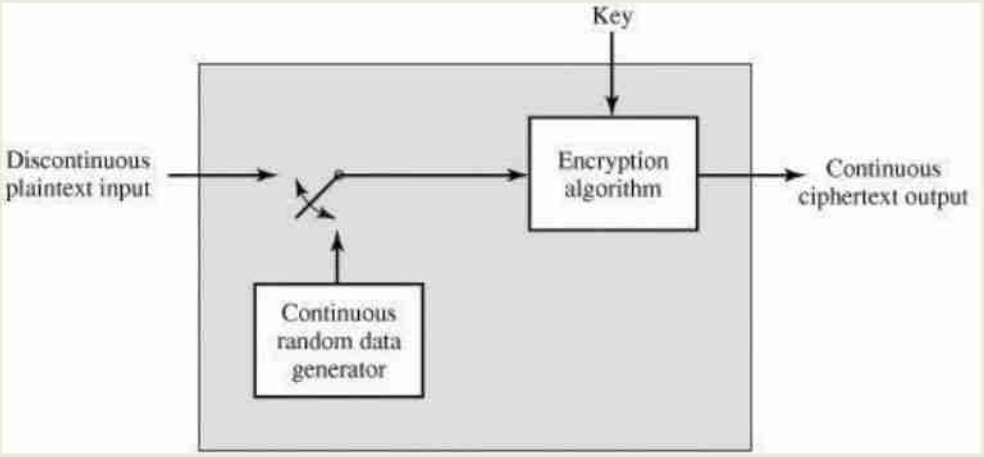


Fig 2.7.1 Traffic-Padding Encryption Device

Traffic padding produces ciphertext output continuously, even in the absence of plaintext. A continuous random data stream is generated. When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and padding and therefore impossible to deduce the amount of traffic.

End-to-End Encryption Approach

Traffic padding is essentially a link encryption function. If only end-to-end encryption is employed, then the measures available to the defender are more limited. For example, if encryption is implemented at the application layer, then an opponent can determine which transport entities are engaged in dialogue. If encryption techniques are housed at the transport layer, then network-layer addresses and traffic patterns remain accessible.

One technique that might prove useful is to pad out data units to a uniform length at either the transport or application level. In addition, null messages can be inserted randomly into the stream. These tactics deny opponent knowledge about the amount of data exchanged between end users and obscure the underlying traffic pattern.

2.8 KEY DISTRIBUTION

A more efficient alternative consists of providing every group member with a single key for securely communicating with a key distribution centre (KDC). This key would be called a master key. When A wants to establish a secure communication link with B, A requests a session key from KDC for communicating with B. In implementation, this approach must address the following issues: – Assuming that A is the initiator of a session-key request to KDC, when A receives a response from KDC, how can A be sure that the sending party for the response is indeed the KDC? – Assuming that A is the initiator of a communication link with B, how does B know that some other party is not masquerading as A? – How does A know that the response received from B is indeed from B and not from someone else masquerading as B? – What should be the lifetime of the session key acquired by A for communicating with B?

Using the key KA for encryption, user A sends a request to KDC for a session key intended specifically for communicating with user B. ˆ The message sent by A to KDC includes A's network address (IDA), B's network address (IDB), and a unique session identifier. The session identifier is a nonce — short for a "number used once" — and we will denoted it N1. The primary requirement on a nonce — a random number — is that it be

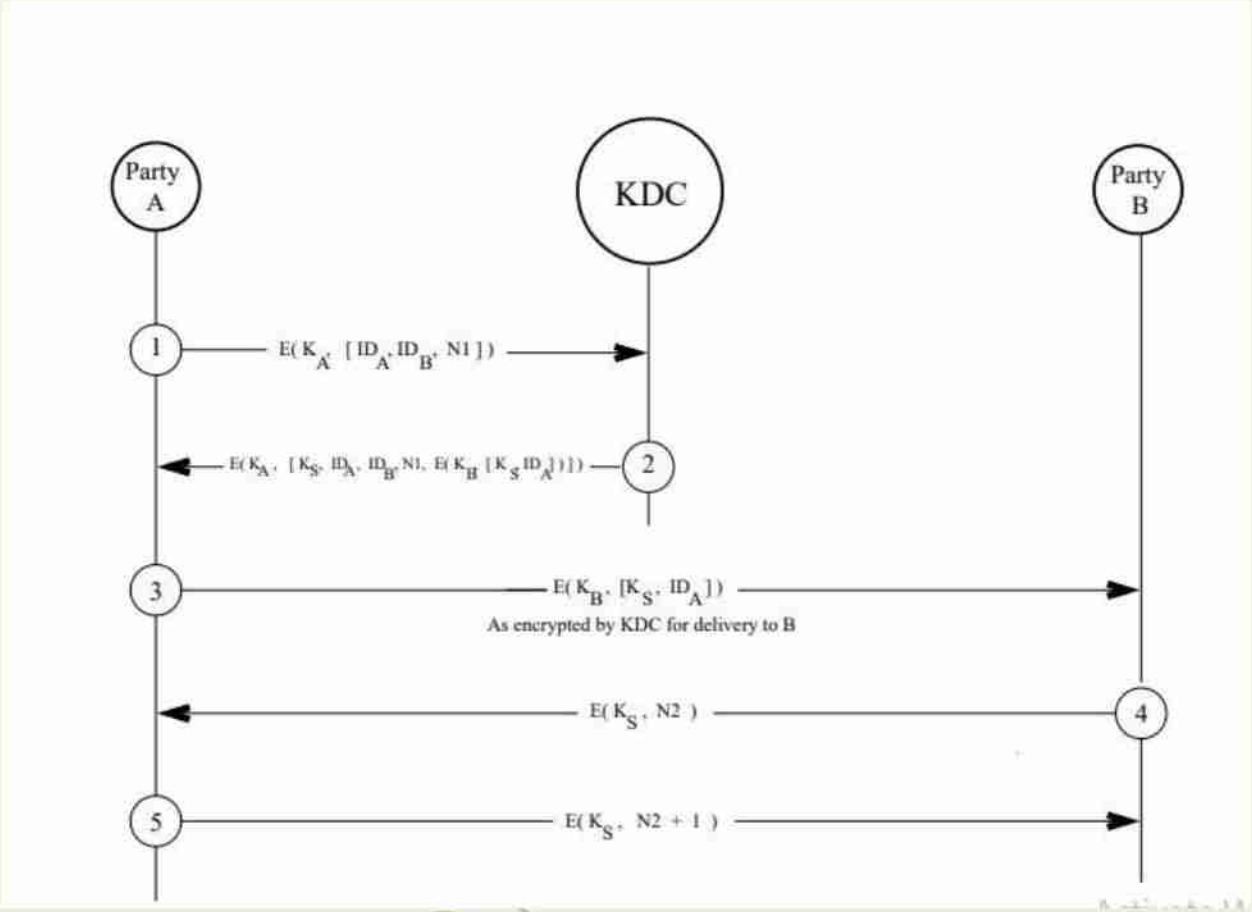


Fig 2.8.1 KDC Distribution

unique to each request sent by A to KDC. The message sent by A to KDC can be expressed in shorthand by E(KA, [IDA, IDB , N1]) where E(., .) stands for encryption of the second-argument data block with a key that is in the first argument. ˆ KDC responds to A with a message encrypted using the key KA. The various components of this message are – The session-key KS that A can use for communicating with B. – The original message received from A, including the nonce used by A. This is to allow A to match the response received from KDC with the request sent. Note that A may be trying to establish multiple simultaneous sessions with B. – A "packet" of information meant for A to be sent to B. This packet of information, encrypted using B's master key KB includes, again, the session key KS, and A's identifier IDA. (Note that A cannot look inside this packet because A does not have access to B's master key KB. We will sometimes refer to this packet of information as a ticket that A receives for sending to B. ˆ The message that KDC sends back to A can be expressed as E(KA, [KS, IDA, IDB , N1, E(KB, [KS, IDA])])

Using the master key KA, A decrypts the message received from KDC. Because only A and KDC have access to the master key KA, A is certain that the message received is indeed from KDC. ˆ A keeps the session key KS and sends the packet intended for B to B. This message is sent to B unencrypted by A. But note that it was previously encrypted by KDC using B's master key KB. Therefore, this first contact from A to B is protected from eavesdropping. ˆ

B decrypts the message received from A using the master key KB. B compares the IDA in the decrypted message with the sender identifier associated with the message received from A. By matching the two, B makes certain that no one is masquerading as A. ˆ

B now has the session key for communicating securely with A. ˆ Using the session key KS, B sends back to A a nonce N2. A responds back with N2 + 1, using, of course, the same session key KS. This serves as a confirmation that the session key KS works for the ongoing session — this requires that what A encrypts with KS be different from what B encrypted with KS. This part of the handshake also ensures that B knows that it did not receive a first contact from A that A is no longer interested in. An additional benefit of this step is that it provides some protection against a replay attack. [A replay attack takes different forms in different contexts. For example, in the situation here, if A was allowed to send back to B the same nonce that it received from the latter, then B could suspect that some other party C posing as A was merely "replaying" back B's message that it had obtained by, say, eavesdropping. In another version of the replay attack, an attacker may repeatedly send an information packet to a victim hoping to elicit from the latter variations on the response that the attacker may then analyze for some vulnerability in the victim's machine. The message sent by B back to A can be expressed as E(KS, [N2]) And A's response back to B as E(KS, [N2 + 1].

2.9 RANDOM KEY GENERATIONS

Random numbers are used by a number of security algorithms for: Nonces (used in authentication protocols) Session key generation (by the KDC or an end system) Key generation for the RSA algorithm. Two requirements: randomness and unpredictability. 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 numbers in the sequence should be uniform; that is, the frequency of occurrence of each of the numbers should be approximately the same.
- Independence: No one value in the sequence can be inferred from the others.

Although there are well-defined tests for determining that a sequence of numbers matches a particular distribution, such as the uniform distribution, there is no such test to "prove" independence. Rather, a number of tests can be applied to demonstrate if a sequence does not exhibit independence. The general strategy is to apply a number of such tests until the confidence that independence exists is sufficiently strong.

In the context of our discussion, the use of a sequence of numbers that appear statistically random often occurs in the design of algorithms related to cryptography. For example, a fundamental

requirement of the RSA public-key encryption scheme discussed in Chapter 9 is the ability to generate prime numbers. In general, it is difficult to determine if a given large number N is prime. A brute-force approach would be to divide N by every odd integer less than $\sqrt{N}$. If N is on the order, say, of 10^{150}, a not uncommon occurrence in public-key cryptography, such a brute-force approach is beyond the reach of human analysts and their computers. However, a number of effective algorithms exist that test the primality of a number by using a sequence of randomly chosen integers as input to relatively simple computations. If the sequence is sufficiently long (but far, far less than $\sqrt{10^{150}}$), the primality of a number can be determined with near certainty. This type of approach, known as randomization, crops up frequently in the design of algorithms. In essence, if a problem is too hard or time-consuming to solve exactly, a simpler, shorter approach based on randomization is used to provide an answer with any desired level of confidence.

Unpredictability

In applications such as reciprocal authentication and session key generation, the requirement is not so much 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. However, as is discussed shortly, true random numbers are seldom used; rather, sequences of numbers that appear to be random are generated by some algorithm. In this latter case, care must be taken that an opponent not be able to predict future elements of the sequence on the basis of earlier elements.

Pseudorandom Number Generators (PRNGs)

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

You may be somewhat uneasy about the concept of using numbers generated by a deterministic algorithm as if they were random numbers. Despite what might be called philosophical objections to such a practice, it generally works. As one expert on probability theory puts it [HAMM91]:

For practical purposes we are forced to accept the awkward concept of "relatively random" meaning that with regard to the proposed use we can see no reason why they will not perform as if they were random (as the theory usually requires). This is highly subjective and is not very palatable to purists, but it is what statisticians regularly appeal to when they take "a random sample"they hope that any results they use will have approximately the same properties as a complete counting of the whole sample space that occurs in their theory.

Linear Congruential Generators

By far, the most 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	$0 \leq m$
$X_0$the starting value, or seed	$0 \leq m$

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 $0 \leq m$.

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, etc.}, which is also clearly unsatisfactory. Of the 32 possible values, only 4 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, etc.}, which increases the period to 8.

We would like m to be very large, so that there is the potential for producing a long series of distinct random numbers. A common criterion is that m be nearly equal to the maximum represent able nonnegative integer for a given computer. Thus, a value of m near to or equal to 2^{31} is typically chosen.

[PARK88] proposes three tests to be used in evaluating a random number generator:

T_1: The function should be a full-period generating function. That is, the function should generate all the numbers between 0 and m before repeating.

T_2: The generated sequence should appear random. Because it is generated deterministically, the sequence is not random. There is a variety of statistical tests that can be used to assess the degree to which a sequence exhibits randomness.

T_3: The function should implement efficiently with 32-bit arithmetic.

With appropriate values of a, c, and m, these three tests can be passed. With respect to T_1 it can be shown that if m is prime and c = 0, then for certain values of a, the period of the generating function is m 1, with only the value 0 missing. For 32-bit arithmetic, a convenient prime value of m is 2^{31} 1. Thus, the generating function becomes

$X_{n+1} = (aX_n) \bmod (2^{31} 1)$

Of the more than 2 billion possible choices for a, only a handful of multipliers pass all three tests. One such value is $a = 7^5 = 16807$, which was originally designed for use in the IBM 360 family of computers [LEWI69]. This generator is widely used and has been subjected to a more thorough testing than any other PRNG. It is frequently recommended for statistical and simulation work (e.g., [JAIN91], [SAUE81]).The strength of the linear congruential algorithm is that if the multiplier and modulus are properly chosen, the resulting sequence of numbers will be statistically indistinguishable from a sequence drawn at random (but without replacement) from the set 1, 2,...,m 1. But there is nothing random at all about the algorithm, apart from the choice of the initial value X_0. Once that value is chosen, the remaining numbers in the sequence follow deterministically. This has implications for cryptanalysis.

If an opponent knows that the linear congruential algorithm is being used and if the parameters are known (e.g., $a = 7^5$, c = 0, m = 2^{31} 1), then once a single number is discovered, all subsequent numbers

are known. Even if the opponent knows only that a linear congruential algorithm is being used, knowledge of a small part of the sequence is sufficient to determine the parameters of the algorithm. Suppose that the opponent is able to determine values for X_0, X_1, X_2 and X_3 Then

$X_1 = (aX_0 + c) \bmod m$

$X_2 = (aX_1 + c) \bmod m$

$X_3 = (aX_2 + c) \bmod m$

These equations can be solved for a, c, and m.

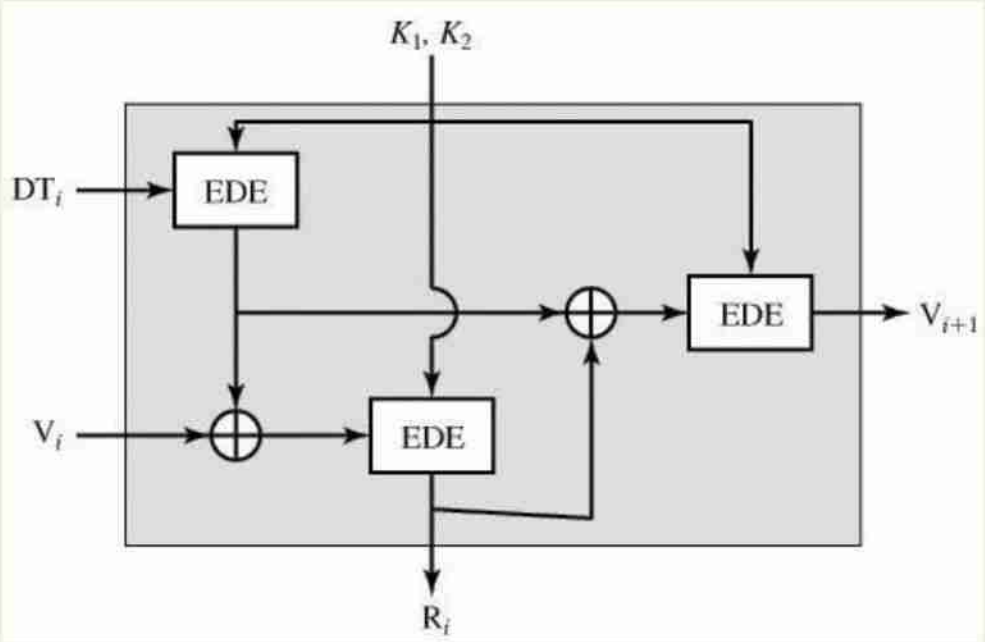


Fig 2.9.1 Pseudorandom Number Generator

Blum Blum Shub Generator

A popular approach to generating secure pseudorandom number is known as the Blum, Blum, Shub (BBS) generator, named for its developers [BLUM86]. It has perhaps the strongest public proof of its cryptographic strength. The procedure is as follows. First, choose two large prime numbers, p and q, that both have a remainder of 3 when divided by 4. That is,

$p \equiv 3 \pmod 4$

This notation, explained more fully in Chapter 4, simply means that (p mod 4) = (q mod 4) = 3. For

example, the prime numbers 7 and 11 satisfy $7 \equiv 11 \equiv 3 \pmod 4$. Let p x q. Next, choose a random number s, such that s is relatively prime to n; this is equivalent to saying that neither p nor q is a factor of s. Then the BBS generator produces a sequence of bits Bi according to the following algorithm:

$X_0 = s^2 \bmod n$ for i = 1 to $X_i = (X_{i1})^2 \bmod n$ $B_i = X_i \bmod 2$

Thus, the least significant bit is taken at each iteration. Table 7.2, shows an example of BBS operation. Here, n = 192649 = 383 x 503 and the seed s = 101355.

i	X_i	B_i
0	20749	
1	143135	1
2	177671	1
3	97048	0
4	89992	0
5	174051	1
6	80649	1
7	45663	1
8	69442	0
9	186894	0
10	177046	0
11	137922	0
12	123175	1
13	8630	0
14	114386	0
15	14863	1
16	133015	1
17	106065	1
18	45870	0
19	137171	1
20	48060	0

The BBS is referred to as a cryptographically secure pseudorandom bit generator (CSPRBG). A CSPRBG is defined as one that passes the next-bit test, which, in turn, is defined as follows [MENE97]: A pseudorandom bit generator is said to pass the next-bit test if there is not a polynomial-time algorithm that, on input of the first k bits of an output sequence, can predict the $(k + 1)^{st}$ bit with probability significantly greater than 1/2. In other words, given the first k bits of the sequence, there is not a practical algorithm that can even allow you to state that the next bit will be 1 (or 0) with probability greater than 1/2. For all practical purposes, the sequence is unpredictable. The security of BBS is based on the difficulty of factoring n. That is, given n, we need to determine its two prime factors p and q.

[7] A polynomial-time algorithm of order k is one whose running time is bounded by a polynomial of order k.

True Random Number Generators

A true random number generator (TRNG) uses a nondeterministic source to produce randomness. Most operate by measuring unpredictable natural processes, such as pulse detectors of ionizing radiation events, gas discharge tubes, and leaky capacitors. Intel has developed a commercially

available chip that samples thermal noise by amplifying the voltage measured across undriven resistors [JUN99]. A group at Bell Labs has developed a technique that uses the variations in the response time of raw read requests for one disk sector of a hard disk [JAKO98]. LavaRnd is an open source project for creating truly random numbers using inexpensive cameras, open source code, and inexpensive hardware. The system uses a saturated CCD in a light-tight can as a chaotic source to produce the seed. Software processes the result into truly random numbers in a variety of formats.

There are problems both with the randomness and the precision of such numbers [BRIG79], to say nothing of the clumsy requirement of attaching one of these devices to every system in an internetwork. Another alternative is to dip into a published collection of good-quality random numbers (e.g., [RAND55], [TIPP27]). However, these collections provide a very limited source of numbers compared to the potential requirements of a sizable network security application. Furthermore, although the numbers in these books do indeed exhibit statistical randomness, they are predictable, because an opponent who knows that the book is in use can obtain a copy.

Skew

A true random number generator may produce an output that is biased in some way, such as having more ones than zeros or vice versa. Various methods of modifying a bit stream to reduce or eliminate the bias have been developed. These are referred to as deskewing algorithms. One approach to deskew is to pass the bit stream through a hash function such as MD5 or SHA-1 (described in Part Two). The hash function produces an n-bit output from an input of arbitrary length. For deskewing, blocks of m input bits, with m ≥

UNIT 3 PUBLIC KEY ENCRYPTION AND KEY MANAGEMENT

Introduction to number theory – Public key cryptography and RSA – Key Management Diffie-hellman Key exchange

1. Introduction

Number Theory

Sending messages in secret has been necessary for thousands of years. If two parties want to communicate without a third party knowing what they are saying, they must correspond in a fashion that the third party couldn't understand even if they saw the message. For example, if ally military leaders want to discuss key battle tactics, they cannot risk their foes intercepting and understanding their messages. This overall idea gave rise to the concept of cryptography. Individuals were enlisted to create ciphers in order to encrypt messages. One famous historical technique is the Caesar Cipher, a primitive method of encryption named after JuliusCaesar. This is an example of a shift cipher, as its idea is to replace each letter with a different letter by shifting the alphabet a specific number of places (e.g. "at" becomes "buy" if the alphabet is shifted by 1). If this was used for the English alphabet, obviously any number but a multiple of 26 would work (as this would shift a letter back to itself). However, since there are only 25 possible ways to shift the alphabet, this was easily broken by codebreakers. Even though more complex ciphers of the same sort are possible, they are often easily broken by frequency analysis, a technique that uses the frequency of letters in words and attempts to match the most common symbols of the encrypted text to the most common letters in the alphabet (e.g. a circle is the most common symbol in the intercepted message and e is the most common letter in the English alphabet, therefore there is a solid chance that the circle represents e).

Following this process, there has been a race between code makers and code breakers for many years. One wants to construct an indecipherable code, and the other will keep attempting to crack the cipher. As math advances, so do the different techniques used to construct ciphers? Overall, this paper will demonstrate that number theory is a crucial component of cryptography by allowing a coherent way of encrypting a message that is also challenging to decrypt. The discussion in this paper follows the set of notes by Evan Dummit.

2.1. **Basic Principles.** We must begin by explaining the math that is useful in cryptography to allow for easier comprehension of specific cryptosystems.

2.1.1. Divisibility and Prime Numbers. Prime numbers are an elementary part ofnumber theory that all readers must understand. First, consider all positive integersbesides 1, e.g. 2, 3, 4, etc. We can divide these numbers into two types: prime numbers and composite numbers. However, prior to going into the definition, we first need to explain the statement "a divides b."

Definition 2.1. For any two integers, we say that "a divides b" or "a b"/if b is divisible by a. In other words, a divides b if b = ac for some integer c.

Example 2.2. 4 | 12, since 12 = 4(3).

Example 2.3. 8 / 56, since 56 = 8(7).

Now, we can explore the idea of prime numbers and composite numbers.

Definition 2.4. An integer n 2 is pr≥ime if the only positive integers that divide n are 1 and n.

Definition 2.5. An integer n is composite if more than two positive integers divide n.

To clarify, every positive integer besides 1 is either prime or composite, as it will always be divisible by at least 1 and itself.

2.1.2. Modular Arithmetic. We will next discuss a part of number theory thathas played a role in a vast array of ciphers: modular arithmetic. To understand modular arithmetic, picture a clock. The maximum number is 12, and no number is larger than that. If one were to reference 5 hours after 12, they would not be referencing 17, as there is no 17 on the clock. They would be talking about 5. This is the idea of modular arithmetic, and this is what we will call "modulo 12."

We define modular arithmetic formally as follows:

Definition 2.6. We say that a ≡ b (mod m) if m divides a − b.

In arithmetic modulo (or "mod") 12, all numbers are equivalent to some number in the ranges 0-11 or 1-12. If we were speak about 20 hours after 6, we would notbe referring to 26, but instead be talking about 2. To reduce a large number to a smaller number modulo 12, we repeatedly subtract 12 from that number until we arrive at a number between 0 and 11. [1]

Additionally, the following are some (but not all) arithmetic rules which still apply:

If a c (mod n) and b d (mod n), then a + b = c + d (mod n) and ab = cd (Mod n).

Example 2.7. (1) 10 is congruent to 2 modulo 4, because 10 2 = 8, which is a multiple of 4.

(2) 127 ≡ 13 (mod 19), because 127 − 13 = 114 = 6 · 19.

The Caesar Cipher from the introduction can be described more succinctly using arithmetic modulo 26. If one wants to shift all letters by 3, then the easy way to accomplish this is the following:

(1) Convert all letters into numbers, with a being 0, b being 1, etc., with zeventually representing 25.

(2) Add 3 to each number, ensuring that one uses modular arithmetic here.For instance, to encrypt c, one uses (2+3) mod 26 = 5 mod 26 = 5.

(3) Convert each number back into a letter. Now, c is represented by f, y isrepresented by b, etc., and z is represented by c. We have our new alphabet.

2.2. Definitions and Theorems to Know.

2.2.1. Definitions and Theorems. We should also express the following definitions and theorems before we begin to discuss cryptography.

Definition 2.8. Two positive integers a and b are relatively prime if there doesnot exist a positive integer c greater than 1 such that c|a and c|b.

Theorem 2.9. Chinese Remainder Theorem: Let m_1, m_2... m_k be relatively prime positive integers such that the greatest common divisor of m_i and m_j is 1 when i = j. Also let a_1, a_2... a_k be arbitrary integers. Then there exists an integer a such that the set of values x satisfying the equations

$x = a_1 \pmod{m_1}$ $x = a_2 \pmod{m_2}$

$x = a_k \pmod{m_k}$

Consists of those integers x congruent to a modulo $m_1m_2...m_k$. essentially, this system of equations has a unique solution modulo $m_1m_2...m_k$.

Definition 2.10. We define ϕ (n) as the number of integers between 1 and n, inclusive, that are relatively prime to n. This function is known as Euler's totient function.

Example 2.11. ϕ (7) = 6.

The numbers between 1 and 7, inclusive that are relatively prime to 7 are 1, 2, 3, 4, 5, and 6. It is important to note here that 7 is prime and ϕ (7) = 6, which is 7 ⊢. More generally, ϕ(p) = p - 1 for every prime number p, as every number lessthan p shares no factors with p besides 1 and is thus relatively prime to p.

Lemma 2.12. If N = PQ where p and p are prime numbers, then ϕ (N) = ϕ (p) ·ϕ (q).

Proof. By the definition, we know that ϕ (N) will tell us the number of integers between 1 and N (inclusive) that are relatively prime to N.We also know thattwo integers are relatively prime if no positive integers greater than 1 divide bothof them. We can picture N as the prime number p which is then multiplied by the other prime q. As a result, N only has one more positive divisor than p (which isq), as q is only divided by 1 and itself. Therefore, only 4 numbers divide N: 1, p,q, and N.

We can conceptually think about ϕ (N) as follows: ϕ (N) will not include p, q, and all the multiples of p and q up to and including N, as those will share a commonfactor with N (either p or q). There are precisely q multiples of p up to N, and there are precisely p multiples of q up to N. Since we only multiplied p and q together once, there is no overlap except for N, which we double counted. Thus, ϕ (N) = N − p − q + 1 = PQ − p − q + 1 = (p − 1) (q − 1) = ϕ (p) · ϕ (q). □

Definition 2.13. The inverse of x modulo m is some number y that satisfies xy

= 1 (mod m). If x has an inverse modulo m, we say that x is a unit modulo m.

Example 2.14. Suppose x = 5 and m = 19. Take y = 4. Then, xy = (5) (4) =20 ≡ 1 (mod 19). Therefore, 5 is a unit modulo 19.

Note that an inverse does not always exist. In fact, the inverse of a modulo m Only exists if a is relatively prime to m.

Definition 2.15. Suppose b is a unit modulo m. The order of b is the smallestinteger e > 0 such that $b^e \equiv 1 \pmod{m}$.

Example 2.16. Consider b = 2 and m = 7.2^1 = 2, which is congruent to 2 mod 7.

$2^2 = 4$, which is congruent to 4 mod 7.
$2^3 = 8$, which is congruent to 1 mod 7.Thus, the order of 2 is 3.

Definition 2.17. We say that a is a primitive root modulo m if a is a unit modulo

M and the order of a is ϕ (m).

Example 2.18. Since 5 is prime, we know that ϕ (5) = 5 ⊥ = 4. Additionally, 3
Is a unit modulo 5 since 7 satisfies 3(7) = 21≡ 1 (mod 5). The order of 3 mod 5
Is 4, since $3^1 = 3$ ∃ (mod 5), $3^2 =$ ∋ ∉ mod 5, $3^3 = 27 \equiv 2$ (mod 5), $3^4 = 81$
1 (mod 5). Thus, since 3 is a unit modulo 5 and the order of 3 is 4, which is ϕ (5),3 is a primitive root modulo 5.

Theorem 2.19. Fermat's Little Theorem: Suppose a is an integer. If p is prime,then $a^{p-1} \equiv 1 \pmod{p}$ if p is prime.

2. Public Key Cryptography

The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography. It is asymmetric, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. Public key schemes are neither more nor less secure than private key (security depends on the key size for both). Public-key cryptography complements rather than replaces symmetric cryptography. Both also have issues with key distribution, requiring the use of some suitable protocol. The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

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

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

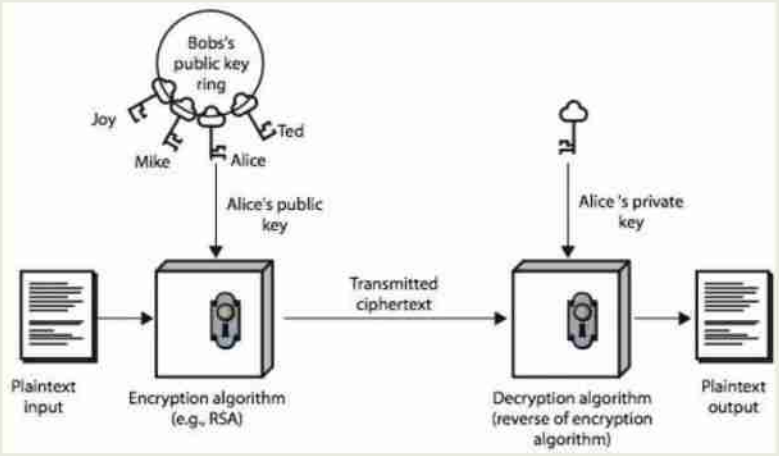
Public-key/two-key/asymmetric cryptography involves the use of **two** keys:

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

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

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

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



The essential steps involved in a public-key encryption scheme are given below: 1.) each user generates a pair of keys to be used for encryption and decryption.

2.) Each user places one of the two keys in a public register and the other key is kept private.

3.) If B wants to send a confidential message to A, B encrypts the message using A's public key.

4.) When A receives the message, she decrypts it using her private key. Nobody else can decrypt the message because that can only be done using A's private key (Deducing a private key should be infeasible).

5.) If a user wishes to change his keys –generate another pair of keys and publish the public one: no interaction with other users is needed. Notations used in Public-key cryptography: The public key of user A will be denoted **KUA**.

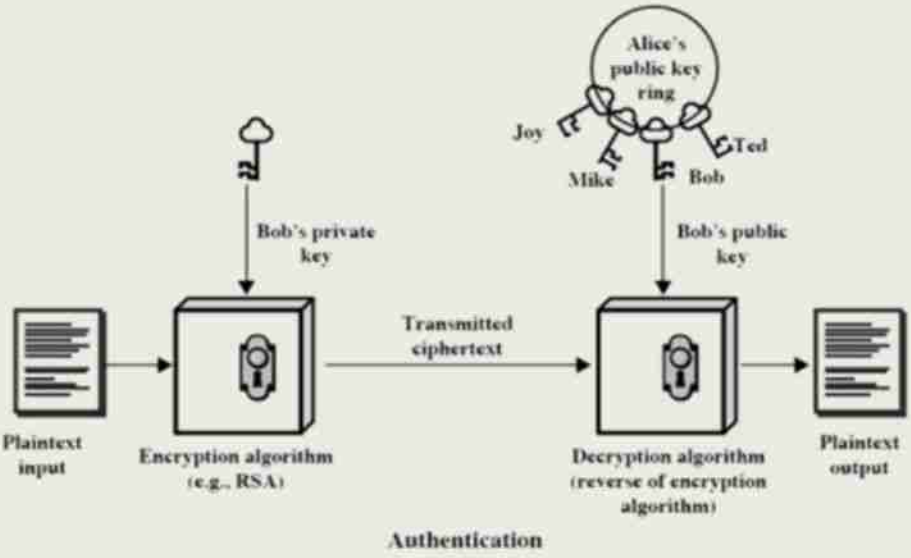
The private key of user A will be denoted **KRA**.

Encryption method will be a function E.

Decryption method will be a function D.

If B wishes to send a plain message X to A, then he sends the crypto text Y=E (KUA, X)
The intended receiver A will decrypt the message: D (KRA, Y) =X

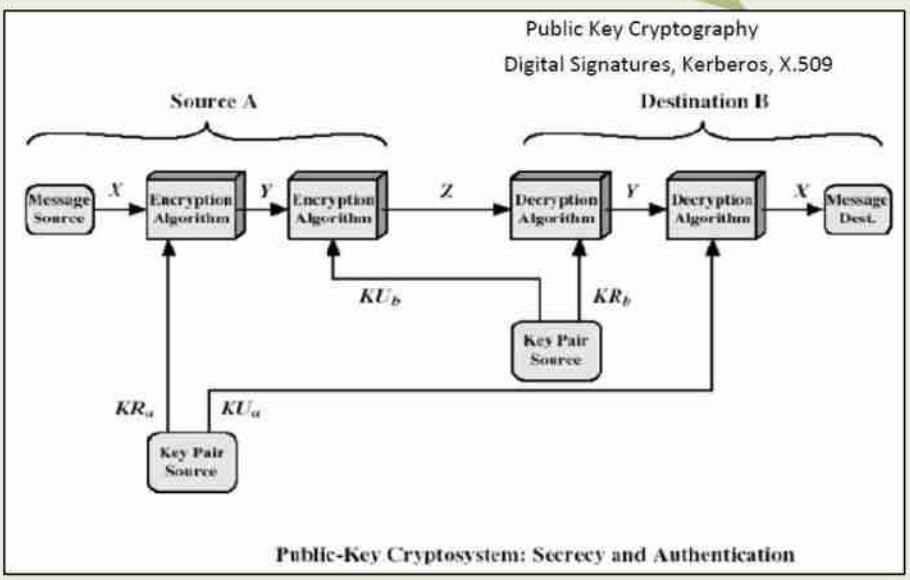
The first attack on Public-key Cryptography is the attack on Authenticity. **An attacker may impersonate user B**: he sends a message E (KUA, X) and claims in the message to be B –A has no guarantee this is so. To overcome this, B will encrypt the message using his private key: Y=E (KRB, X). Receiver decrypts using B's public key KRB. This shows the authenticity of the sender because (supposedly) he is the only one who knows the private key. The entire encrypted message serves as a digital signature. This scheme is depicted in the following figure:



Authentication

Public Key Cryptography

Digital Signatures, Kerberos, X.509



Public-Key Cryptosystem: Secrecy and Authentication

But, a drawback still exists. Anybody can decrypt the message using B's public key. So, secrecy or confidentiality is being compromised. One can provide both authentication and confidentiality using the public-key scheme twice:

B encrypts X with his private key: Y=E (KRB, X) B encrypts Y with A's public key: Z=E (KUA, Y)
A will decrypt Z (and she is the only one capable of doing it): Y=D (KRA, Z)

A can now get the plaintext and ensure that it comes from B (he is the only one who knows his private key): decrypt Y using B's public key: X=E (KUB, Y).

Applications for Public-Key Cryptosystems:

1.) **Encryption/decryption**: sender encrypts the message with the receiver's public key.

2.) **Digital signature**: sender "signs" the message (or a representative part of the message) using his private key

3.) **Key exchange**: two sides cooperate to exchange a secret key for later use in a secret-key cryptosystem.

The main requirements of Public-key cryptography are:

1. Computationally easy for a party B to generate a pair (public key Kobe, privatekey KRb).
2. Easy for sender A to generate cipher text:
3. Easy for the receiver B to decrypt cipher text using private key:
4. Computationally infeasible to determine private key (KRb) knowing public key (KUb)
5. Computationally infeasible to recover message M, knowing KUb and cipher text C
6. either of the two keys can be used for encryption, with the other used for decryption:

M= DKRb [EKUb (M)] =DKUb [EKRb (M)]

Easy is defined to mean a problem that can be solving polynomial time as a function of input length. A problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size. Public-key cryptosystems usually rely on difficult math functions rather than -P networks as classical cryptosystems. **One-way function** is one, easy to calculate in one direction, infeasible to calculate in the other direction (i.e., the inverse is infeasible to compute). **Trap-door function** is a difficult function that becomes easy if some extra information is known. Our aim to find a **trap-door one-way function**, which is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known.

Security of Public-key schemes:

1. Like private key schemes brute force **exhaustive search** attack is always theoretically possible. But keys used are too large (>512bits).
2. Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalysis) problems. More generally the **hard** problem is known, it's just made too hard to do in practice.
3. Requires the use of **very large numbers,** hence is **slow** compared to private key schemes.

3. RSA Algorithm

RSA is the best known, and by far the most widely used general public key encryption algorithm, and was first published by Rivest, Shamir & Adleman of MIT in 1978 [RIVE78]. Since that time RSA has reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption. The RSA scheme is a block cipher in which the plaintext and the ciphertext are integers between 0 and n-1 for some fixed n and typical size for n is 1024 bits (or 309 decimal digits). It is based on exponentiation in a finite (Galois) field over integers modulo a prime, using large integers (Eg. 1024 bits). Its security is due to the cost of factoring large numbers. RSA involves a public-key and a private-key where the public key is known to ll and is used to encrypt data or message. The data or message which has been encrypted using a public key can only be decrypted by using its corresponding private-k y. Each user generates a key pair i.e. public and private key using the following steps:

- each user selects two large primes at random - p, q
- compute their system modulus n=p.q
- calculate ø(n), where **ø(n)=(p-1)(q- 1)**
- selecting at random the encryption key e, where 1<e<ø(n),and **gcd(e,ø(n))=1**
- solve following equation to find decryption key d: **e.d=1 mod ø(n)** and 0≤d≤n
- publish their public encrption key: **KU={e,n}**
- keep secret private decryption key**: KR={d,n}**

Both the sender and receiver must know the values of n and e, and only the receiver knows the value of d. Encryption and Decryption are done using the following equations. To encrypt a message M the sender:

- obtains **public key** of recipient KU={e,n}
- computes: **C=Me mod n**, where 0≤M<n To decrypt the ciphertext C the owner:
- uses their private key KR={d,n}
- computes: **M=Cd mod n = (Me) d mod n = Med mod n**

For this algorithm to be satisfactory, the following requirements are to be met.

a) It's possible to find values of e, d, n such that **Med = M mod n** for all M<n

b) It is relatively easy to calculate Me and C for all values of M < n.

c) It is impossible to determine d given e and n

The way RSA works is based on Number theory: **Fermat's little theorem**: if **p** is prime and **a** is positive integer not divisible by **p**, then **ap-1 ≡ 1 mod p. Corollary**: For any positive integer **a** and prime **p**, **ap ≡ a mod p.**

Fermat's theorem, as useful as will turn out to be does not provide us with integers d,e we are looking for –Euler's theorem (a refinement of Fermat's) does. Euler's function associates to any positive integer **n,** a number φ**(n)**: the number of positive integers smaller than **n** and relatively prime to **n.** For example, φ**(37) = 36** i.e. φ**(p) = p-1** for any prime **p.** For any two primes **p,q**, φ**(pq)=(p-1)(q-1). Euler's theorem**: for any relatively prime integers a,n we have **a**φ**(n)≡1 mod n. Corollary: F**or ny integers a,n we have **a**φ**(n)+1≡a mod n Corollary:** Let p,q be two odd primes and n=pq. Then: φ(n)=(p-1)(q-

1) For any integer m with 0<m<n, m(p-1)(q-1)+1 ≡ m mod n For any integers k,m with 0<m<n, mk(p-1)(q-1)+1 ≡ m mod n Euler's theorem provides us the numbers d, e such that Med=M mod n. We have to choose d,e such that ed=kφ(n)+1, or equivalently, d≡e-1mod φ(n)

An example of RSA can be given as, Select primes: p=17 & q=11 Compute $n = pq$

=17×11=187
Compute ø (*n*) = (*p*–1) (*q*-1) =16×10=160 Select e: gcd (e, 160) =1; choose *e*=7
Determine d*: de*=1 mod 160 and *d* < 160 Value is d=23 since 23×7=161= 10×160+1 Publish public key KU= {7,187}
Keep secret private key KR= {23,187} Now, given message M = 88 (nb. 88<187) encryption: C = 887 mod 187 = 11
Decryption: M = 1123 mod 187 = 88

Another example of RSA is given as,

Let **p = 11, q = 13, e = 11, m = 7**

n = pq i.e. n= 11*13 = 143

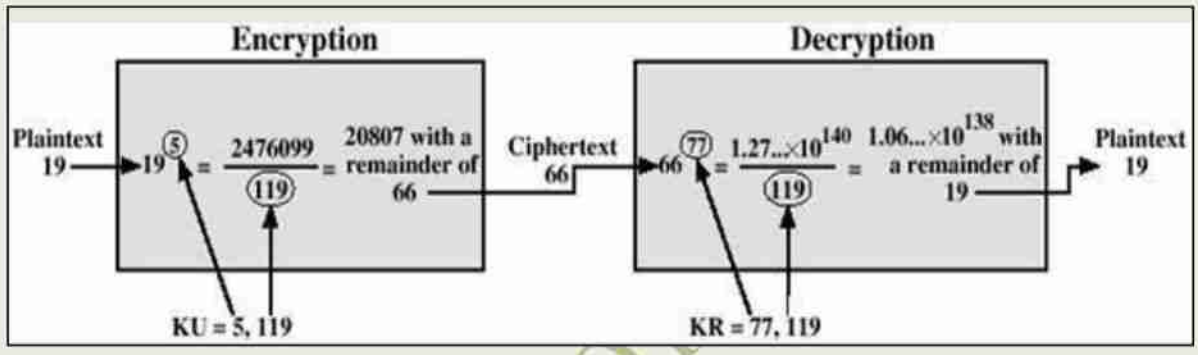
ø (n)= (p-1)(q-1) i.e. (11-1)(13-1) = 120

e.d=1 mod ø(n) i.e. 11d mod 120 = 1 i.e. (11*11) mod 120=1; so d

= 11 public key :{11,143} and private key: {11,143}

C=Me mod n, so ciphertext = 711mod143 = 727833 mod 143; i.e. C = 106

M=Cd mod n, plaintext = 10611 mod 143 = 1008 mod 143; i.e. M = 7



For RSA key generation,

Users of RSA must:

- Determine two primes at random - p, q
- select either e or d and compute the other
- earns must be sufficiently large
- typically guess and use probabilistic test

Security of RSA

There are three main approaches of attacking RSA algorithm.

Brute force key search (infeasible given size of numbers) As explained before, involves trying all possible private keys. Best defense is using large keys.
Mathematical attacks (based on difficulty of computing ø(N), by factoring modulus N) There are several approaches, all equivalent in effect to factoring the product of two primes. Some of them are given as:

- factor N=p.q, hence find ø(N) and then d
- determine ø(N) directly and find d
- find d directly

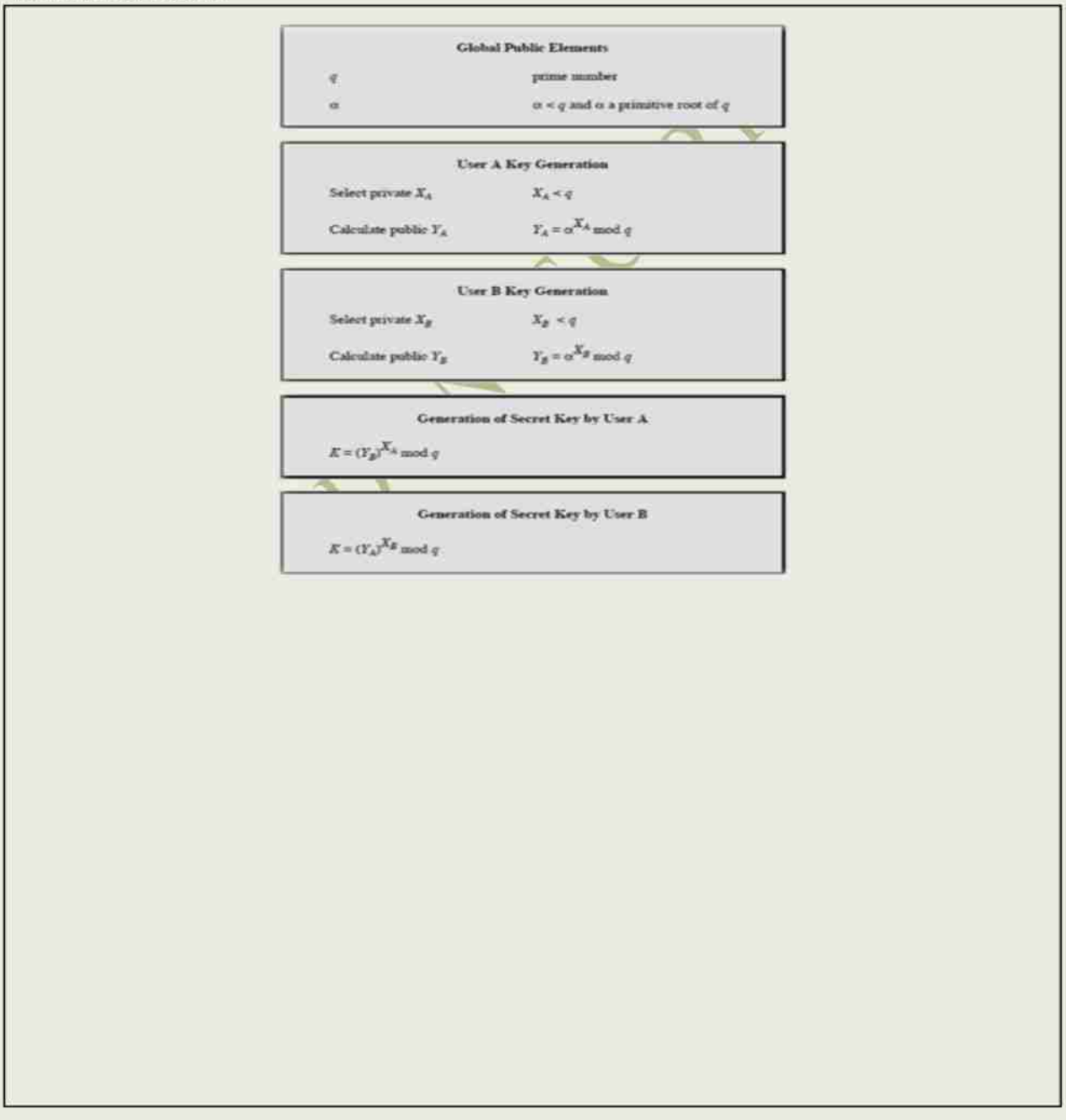
The possible defense would be using large keys and also choosing large numbers for p and q, which should differ only by a few bits and are also on the order of magnitude 1075 to 10100. And gcd (p-1, q-1) should be small.

4. Diffie-Hellman Key Exchange

Diffie-Hellman key exchange (**D-H**) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher$_{Media}$.TheD-Hlgorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

First, a primitive root of a prime number p, can be fined as one whose powers generate all the integers from 1 to p-1. If a is a primitive root of the prime number p, then the numbers, **a mod p, a2 mod p,..., ap-1 mod p**, are distinct and consist of the integers from 1 through p 1 in some permutation.
For any integer **b** and a primitive root **a** of prime number **p**, we can find a unique exponent

i such that $b \equiv a^i \pmod{p}$ where $0 \leq i \leq (p\ 1)$.The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as **dloga,p (b).** The algorithm is summarized below:



For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q. suppose the users A and B wish to exchange a key. User A selects a random integer XA < q and computes YA = αXA mod q. Similarly, user B independently selects a random integer XA < q and computes YB = αXB mod q. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as K = (YB) XA mod q and user B computes the key as K = (YA) XB mod q. These two calculations produce identical results.

Discrete Log Problem

The (discrete) exponentiation problem is as follows: Given a base a, an exponent b and a modulus p, calculate c such that $ab \equiv c \pmod{p}$ and $0 \leq c < p$. It turns out that this problem is fairly easy and can be calculated "quickly" using fast-exponentiation. The discrete log problem is the inverse problem: Given a base a, a result c ($0 \leq c < p$) and a modulus p,Calculate the exponent b such that $ab \equiv c \pmod{p}$. It turns out that no one has found a quick way to solve this problem With DLP, if P had 300 digits, Xa and Xb have more than 100 digits, it would take longer than the life of the universe to crack the method.

Examples for D-H key distribution scheme:

1) Let p = 37 and g = 13.

Let Alice pick a = 10. Alice calculates 1310 (mod 37) which is 4 and sends that to Bob. Let Bob pick b = 7. Bob calculates 137 (mod 37) which is 32 and sends that to Alice. (Note: 6 and 7 are secret to Alice and Bob, respectively, but both 4 and 32 are known by all.)

2) Let p = 47 and g = 5. Let Alice pick a = 18. Alice calculates 518 (mod 47) which is 2 and sends that to Bob. Let Bob pick b = 22. Bob calculates 522 (mod 47) which is 28 and sends that to Alice.

Man-in-the-Middle Attack on D-H Protocol

Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

1. Darth prepares for the attack by generating two random private keys XD1 and XD2 and then computing the corresponding public keys YD1 and YD2.
2. Alice transmits YA to Bob.
3. Darth intercepts YA and transmits YD1 to Bob. Darth also calculates K2 = (YA) XD2mod q.
4. Bob receives YD1 and calculates K1 = (YD1) XE mod q.
5. Bob transmits XA to Alice.
6. Darth intercepts XA and transmits YD2 to Alice. Darth calculates K1 = (YB) XD1 mod q.
7. Alice receives YD2 and calculates K2 = (YD2) XA mod q.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message M: E (K2, M).

2. Darth intercepts the encrypted message and decrypts it, to recover M.

3. Darth sends Bob E(K1, M) or E (K1, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public- key certificates.

Elliptic Curve Cryptography (Ecc)

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested independently by N al Koblitz and Victor S. Miller in 1985. The principal attraction of ECC compared to RSA is that it appears to offer equal security for a far smaller bit Sky upsize, thereby reducing the processing overhead.

Elliptic Curve over GF (p)

Let GF (p) be a finite field, > 3, and let a, b

 GF(p) are constant such that ***4a3 + 27b2 ≡ 0 (mod p).*** An elliptic curve, E(a,b)(GF(p)),

is defined as the set of points (x,y) cGF(p) * GF(p) which satisfy the equation

y2 ≡ x3 + ax + b (mod p), together with a special point, O, *called the point at infinity*. Let P and Q be two points on E(a,b)(GF(p)) and O is the point at infinity.

- P+O = O+P = P
- If P = (x1, y1) then -P = (x1,-y1) and P + (-P) = O.
- If P = (x1, y1) and Q = (x2, y2), and P and Q are not O.

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \; for \; \_ x_1 \neq x_2 \\ \dfrac{3x_1^{\,2} + a}{2y_1} \; for \; \_ x_1 = x_2 \end{cases}$$

Then P +Q = (x3 ,y3) where

X3 = λ 2 - x1 - x2

y3 = λ (x1 - x3) - y1 and

λ = (y2-y1)/(x2-x1) if P ≠ Q

λ = (3x12+a)/ 2y1 if P = Q

An elliptic curve may be defined over any finite field GF (*q*). For GF (2*m*), the curve has a different form: - ***y2 + my = x3 + ax2 + b,*** *where b! =0.*

Cryptography with Elliptic Curves

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple additions are the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, some kind of hard problem such as discrete logarithm or factorization of prime numbers is needed. Considering the equation, Q=kP, where Q,P are points in an elliptic curve, it is “easy” to compute Q given k,P , but “hard” to find k given Q,P. This is known as the elliptic curve logarithm problem. K could be so large as to make brute-force fail.

Pick a prime number p= 2180 and elliptic curve parameters and b for the equation

$y2 \equiv x3 + ax + b \pmod p$ which defines the elliptic group of points Ep(a,b). Select generator point G=(x1,y1) in Ep(a,b) such that the smallest value for which nG=O

ECC Key Exchange be a very large prime number. Ep(a,b) and G are parameters of the cryptosystem known to all participants. The following steps take place:

- A & B select private keys nA<n, nB<n
- compute public keys: PA=nA×G, PB=nB×G
- Compute shared key: K=nA×PB, K=nB×PA {same since K=nA×nB×G }

ECC Encryption/Decryption As with key exchange system, an encryption/decryption system requires a point G and and elliptic group Ep (a,b) as parameters. First thing to be done is to encode the plaintext message m to be sent as an x-y point **Pm**. Each user chooses private key nA<n and computes public key PA=nA×G. To encrypt and send a message to Pm to B, A chooses a random positive integer k and produces the ciphertext **Cm** consisting of the pair of points **Cm={kG, Pm+kPb}.** Here, A uses B's public key.

To decrypt the cipher text, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point **Pm+*k*Pb – nB(*kG*) = Pm+*kPb*(nB*G*) – nB(*kG*) = Pm** A has masked the message **Pm** by adding ***k*Pb** to it. Nobody but A knows the value of k, so even though **Pb** is a public key, nobody can remove the mask k**Pb.** For an attacker to recover the message, he has to compute k given G and kG, which is assumed hard.

Security of ECC To **protect** a 128 bit AES key it would take a RSA Key Size of 3072 bits whereas an ECC Key Size of 256 bits.

Computational Effort for Cryptanalysis of Elliptic Curve Cryptography Compared to RSA

Key Size	MIPS-Years
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

(a) Elliptic Curve Logarithms using the Pollard rho Method

Key Size	MIPS-Years
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

(b) Integer Factorization using the General Number Field Sieve

Hence for similar security ECC offers significant computational advantages.

Applications of ECC:

- Wireless communication devices
- Smart cards
- Web servers that need to handle many encryption sessions
- Any application where security is needed but lacks the power, storage and computational power that is necessary for our current cryptosystems

Key Management

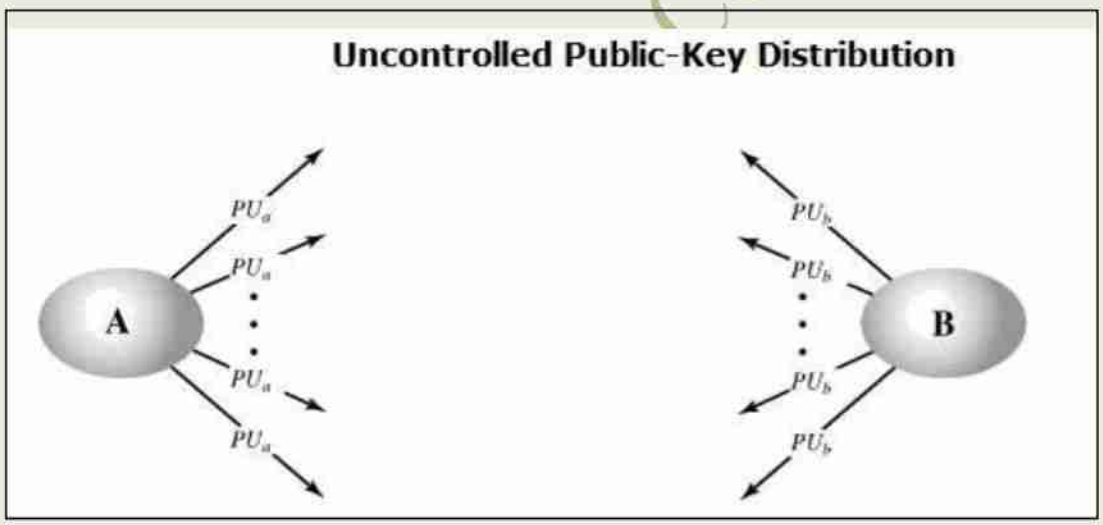
One of the major roles of public-key encryption has been to address the problem of key distribution. Two distinct aspects to use of public key encryption are present.

- The distribution of public keys.
- Use of public-key encryption to distribute secret keys.

Distribution of Public Keys The most general schemes for distribution of public keys are given below

Public Announcement of Public Keys

Here any participant can send his or her public key to any other participant or broadcast the key to the community at large. For example, many PGP users have adopted the practice of appending their public key to messages that they send to public forums.

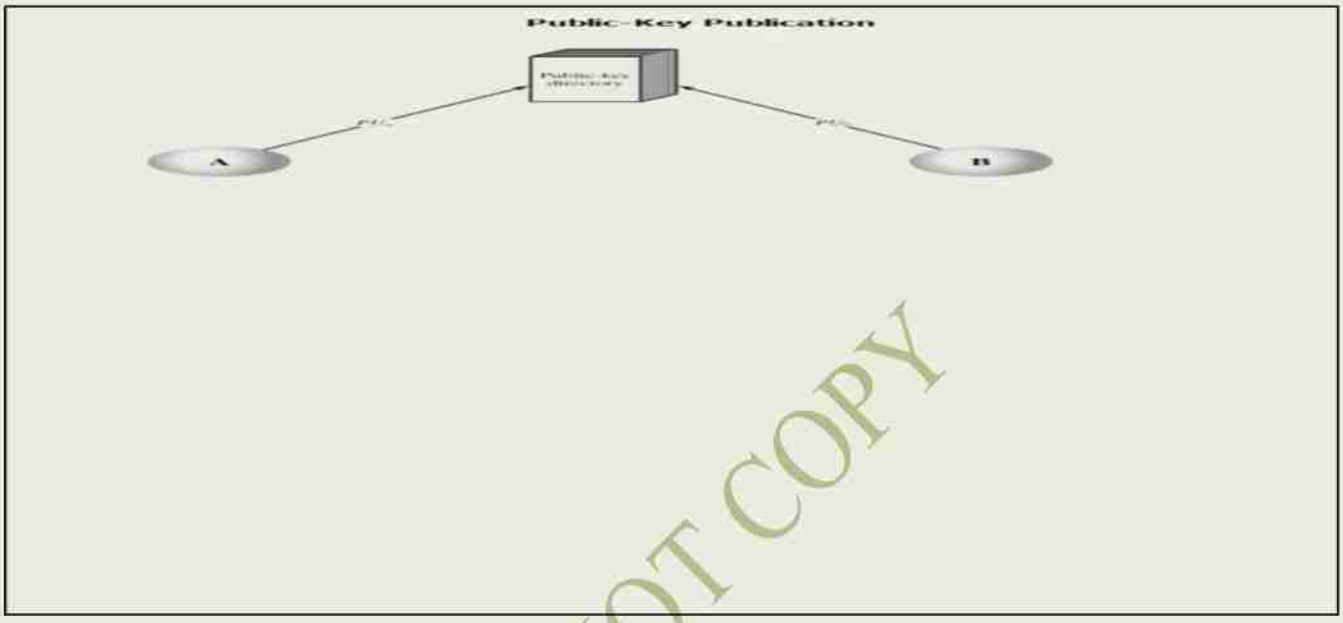


Though this approach seems convenient, it has a major drawback. Anyone can forge such a public announcement. Some user could pretend to be user and send a public key to another participant or broadcast such a public key. Until the time when A discovers about the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization. It includes the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.



3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory. This scheme has still got some vulnerability. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively a out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Or else, the adversary may tam er with the records kept by the authority.

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. This scenario assumes the existence of a public authority (whoever that may be) that maintains a dynamic directory of public keys of all users. The public authority has its own (private key, public key) that it is using to communicate to users. Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. For example, consider that Alice and Bob wish to communicate with each other and the following steps take place and are also shown in the figure below:

1.) Alice sends a **timestamped** message to the central authority with a request for Bob's public key (the time stamp is to mark the moment of the request)

2.) The authority sends back a message encrypted w th its private key (for authentication) – message contains Bob's public k y and the original message of Alice – this way Alice knows this is not a reply to an old request;

3.) Alice starts the communication to Bob by sending him an encrypted message containing her identity IDA and a nonce N 1 (to identify uniquely this transaction)

4.) Bob requests Alice's public key in the same way (step 1)

5.) Bob acquires Alice's public key in the same way as Alice did. (Step-2)

6.) Bob replies to Alice by sending an encrypted message with N1 plus a new generated nonce N2 (to identify uniquely the transaction)

7.) Alice replies once more encrypting Bob's nonce N2 to assure bob that its correspondent is Alice

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

Public-Key Certificates

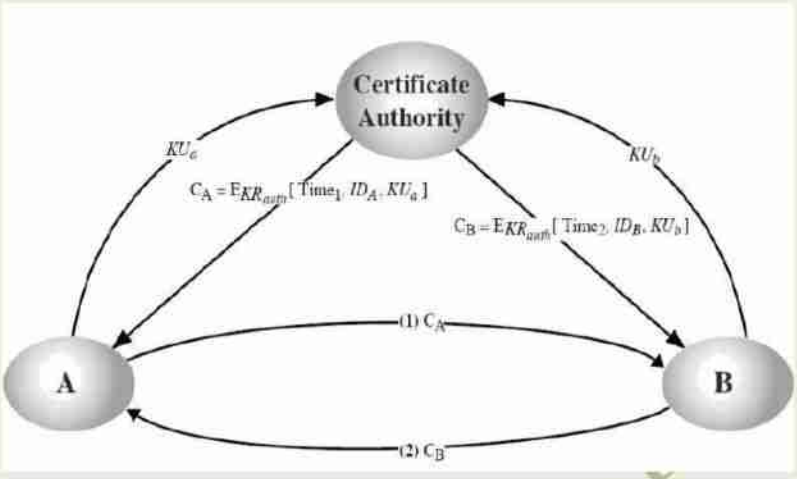
The above technique looks attractive, but still has some drawbacks. For any communication between any two users, the central authority must be consulted by both users to get the newest public keys i.e. the central authority must be online 24 hours/day. If the central authority goes offline, all secure communications get to a halt. This clearly leads to an undesirable bottleneck. A further improvement is to use certificates, which can be used to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. A certificate binds an **identity** to **public key**, with all contents **signed** by a trusted Public-Key or Certificate Authority (CA). A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate. Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. This certificate issuing scheme does have the following requirements:

1. Any participant can read a certificate to determine the n me and public key of the certificate's owner.

2. Any participant can verify that the certificate originate from the certificate authority and is not counterfeit.

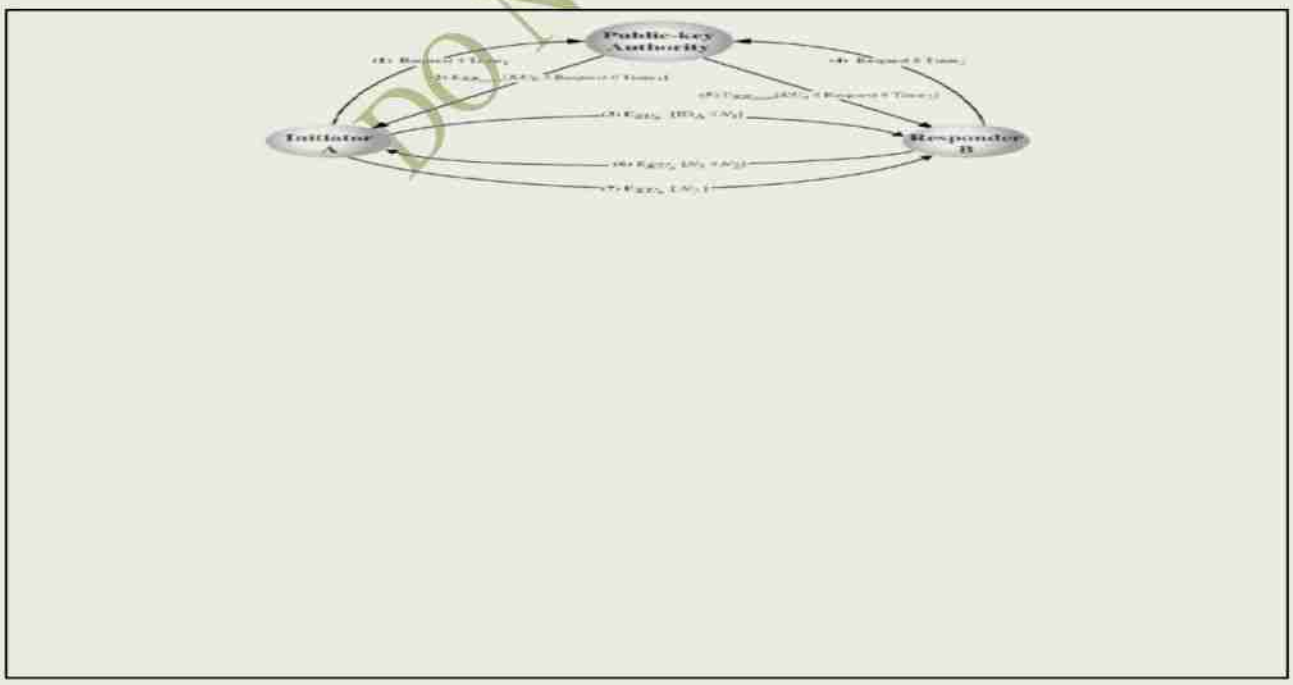
3. Only the certificate authority can create and update certificates.

4. Any participant can verify the currency of the certificate.



Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

CA = E(PRauth, [T||IDA||PUa]) where PRauth is the private key used by the authority



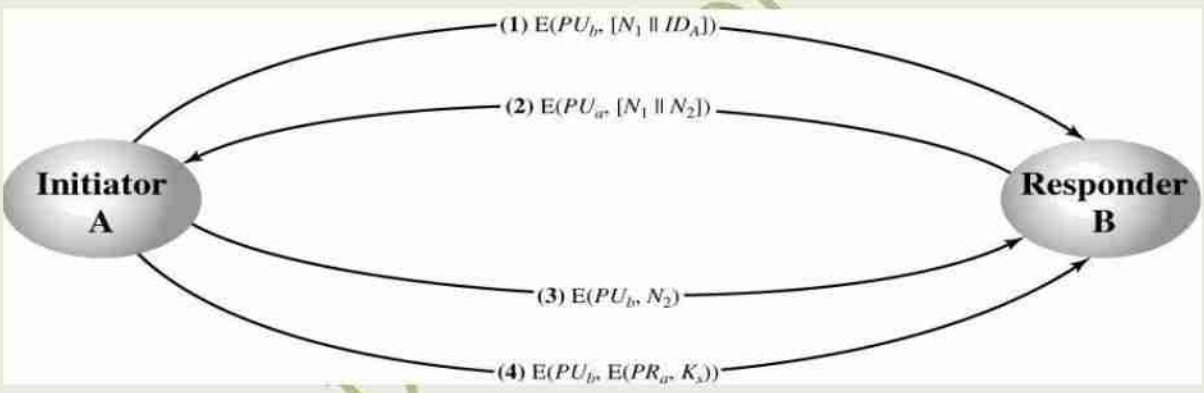
and T is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows: D(PUauth, CA) = D(PUauth, E(PRauth,

[T||IDA||PUa])) = (T||IDA||PUa) The recipient uses the authority's public key, PUauth to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements IDA and PUa provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate. Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can re d those messages. In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired.

One scheme has becomeSkyupsuniversallyaccepted for formatting public-key certificates: the

X.509 standard. X.509 certificates are ed in most network security applications, including IP security, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME. **Secret Key Distribution With Confidentiality And Authentication**

It is assumed that A and B have exchanged public keys by one of the schemes described earlier. Then the following steps occur:



1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) and a nonce (N1), which is used to identify this transaction uniquely.
2. B sends a message to A encrypted with PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2) Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.
3. A returns N2 encrypted using B's public key, to assure B that its correspondent is A.A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it. B computes D (PUa, D(PRb, M)) to recover the secret key. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

Part-A			
Qno	**Questions**	**Competence**	**BT Level**
1.	Define Discrete Logarithm	Understand	BTL 2
2.	State the use of Fermat's Theorem	Remember	BTL 1
3.	Find 11^7 mod 13.	Apply	BTL 3
4.	State Euler's Theorem	Remember	BTL 1
5.	Define Finite Field	Understand	BTL 2
6.	Define Diffusion	Understand	BTL2
7.	Write down the difference between the public key and private key cryptosystems	Create	BTL6
8.	Is it possible to use the DES algorithm to generate message authentication code? Justify.	Analysis	BTL4
9.	Mention the application of public key cryptography.	Evaluate	BTL5
10	State the difference between conventional encryption and public-key encryption.	Analysis	BTL4
Part-B			
Q.No	**Questions**	**Competence**	**BT Level**
1.	Explain briefly about Fermat's and Euler's theorem	Understand	BTL1
2.	In a public key system using RSA, you intercept the cipher text C=10 sent to a user whose public key is e=5, n=35. What is the plain text? Explain the above problem with an algorithm description	Analyse	BTL4
3.	Describe Chinese Reminder theorem.	Understand	BTL1
4.	Explain Diffie-Hellman key exchange algorithm with an example. Consider a Diffie-Hellman scheme with a common prime q=353 and a primitive root α=3.	Apply	BTL3

	Users A and B have private keys X_A=17 and X_B=21 respectively. What is the shared secret key K1 and K2?		
5.	Demonstrate encryption and decryption for the RSA algorithm parameters: p=3, q=11, e=7, d=? , M=5.	Apply	BTL3
6.	Users A and B use the Diffie-Hellman Key exchange technique with a common prime q=71 and a primitive root $\alpha = 7$. If the user A has private key X_A=5, what is A's public key Y_A?	Evaluate	BTL5
7.	Explain Diffie-Hellman Key exchange algorithm with its merits and demerits.	Understand	BTL1

Authentication requirements – Authentication functions – message authentication codes – Hash functions – Security of hash functions and MAC'S – MD 5 (Message Digest Algorithm) – HMAC. Digital Signatures and authentication protocols: Digital Signatures – Authentication protocols – Digital Signature Standard – Kerbews – X.509 Authentication Service

I. AUTHENTICATION REQUIREMENTS

In the context of communication across a network, the following attacks can be identified:

Disclosure – releases of message contents to any person or process not possessing the appropriate cryptographic key.

Traffic analysis – discovery of the pattern of traffic between parties.

Masquerade – insertion of messages into the network fraudulent source.

Content modification – changes to the content of the 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.

Source repudiation – denial of transmission of message by source.

Destination repudiation – denial of transmission of message by destination.

Measures to deal with first two attacks are in the realm of message confidentiality. Measures to deal with 3 through 6 are regarded as message authentication. Item 7 comes under digital signature and dealing with item 8 may require a combination of digital signature and a protocol to counter this attack.

II. AUTHENTICATION FUNCTIONS

Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there may be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower layer function is then used as primitive in a high **the different types of functions** that may be used to produce an **authenticator**

are as follows:

Message encryption – the cipher text of the entire message serves as its authenticator.

Message authentication code (MAC) – a public function of the message and a secretkey that produces a fixed length value serves as the authenticator.

Hash function – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

Message encryption

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

er-layer authentication protocol that enables a receiver to verify the authenticity of a message.

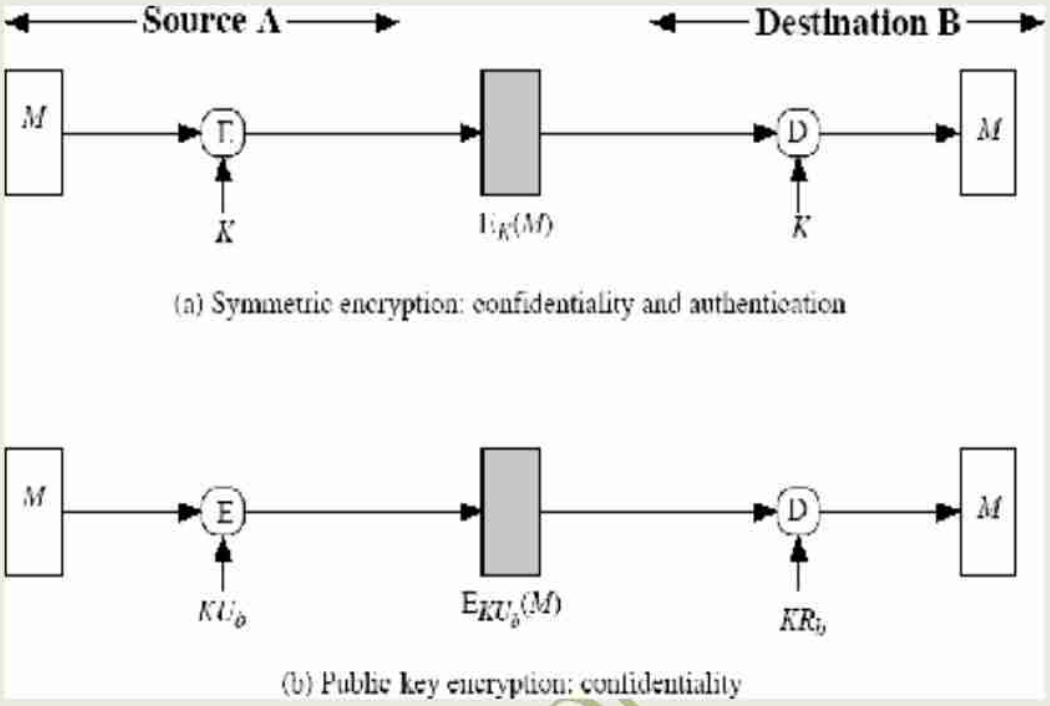


Figure 1 Authentication Functions

Suppose the message can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message. One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function. We could, for example, append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption

'A' prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the result as a message with an appended FC. B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic.

In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext (encrypted message).

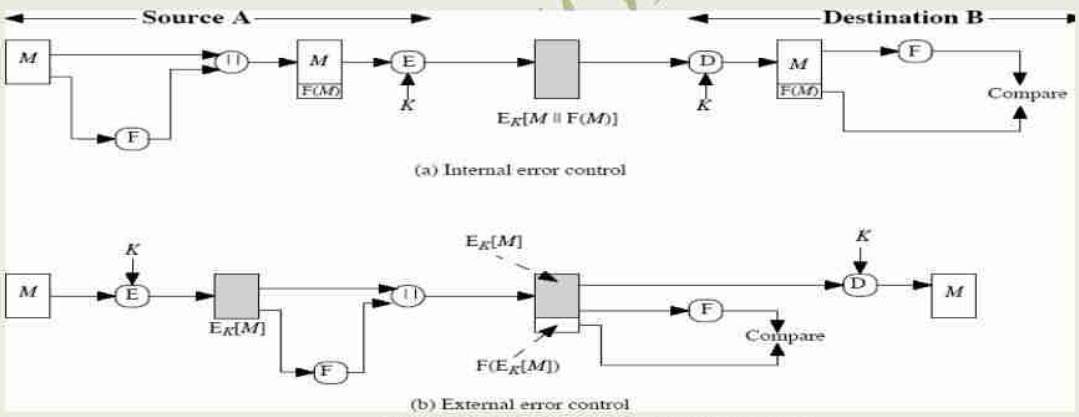


Figure 2 Frame Check Sequence

III. MESSAGE AUTHENTICATION CODE (MAC)

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message to B, it calculates the MAC as a function of the message and the key.

MAC = $C_K(M)$ Where M – input message

C – MAC function
K – Shared secret key

+MAC - Message Authentication Code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC. The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic.

A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many- to-one function.

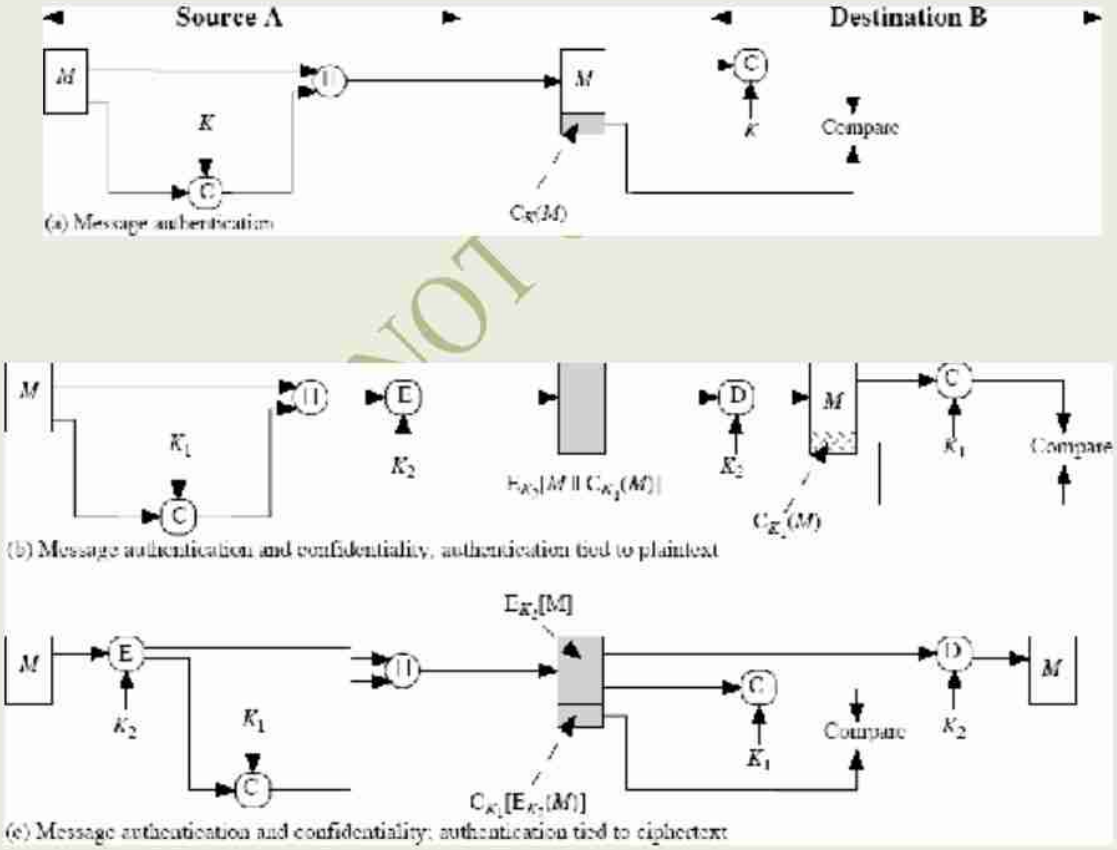


Figure 3 MAC Encryption

Requirements for MAC:

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute- force attack using all possible keys.
On average, such an attack will require $2^{(k-1)}$ attempts for a k-bit key.
In the case of a MAC, the considerations are entirely different. Using brute-force methods, how would an opponent attempt to discover a key?

If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the MAC size. Then, given a known M1 and MAC1, with MAC1 = CK (M1), the cryptanalyst can perform MACi
= CKi (M1) for all possible key values Ki.
At least one key is guaranteed to produce a match of MACi = MAC1.
Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

Round 1

Given: M1, MAC1 = CK(M1)
Compute MACi = CKi (M1) for all 2^k

keys Number of matches $\approx 2^{(k-n)}$

Round 2

Given: M2, MAC2 = CK(M2)
Compute MACi = CKi (M2) for the $2^{(k-n)}$ keys resulting from Round

1Number of matches $\approx 2^{(k-2xn)}$

and so on. On average, a rounds will be needed if k = a x n. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} possibilities. The third round should produce only
a single key, which must be the one used by the sender.

If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match.

Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.

Consider the following MAC algorithm. Let M = (X1||X2||...||Xm) be a message that is treated as a concatenation of 64-bit blocks Xi. Then define

$\Delta(M) = X1 \oplus X2 \oplus \dots \oplus Xm$

$Ck(M) = Ek(\Delta(M))$

where is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 2^{56} encryptions. But the opponent can attack the system by replacing X1 through Xm-1 with any desired values Y1 through Ym-1 and replacing Xm with Ym where Ym is calculated as follows:

$Ym = Y1 \oplus Y2 \oplus \dots \oplus Ym1 \oplus \Delta(M)$

The opponent can now concatenate the new message, which consists of Y1 through Ym, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 x (m-1) bits can be fraudulently inserted.

Then the MAC function should satisfy the following requirements: The MAC function should

havethe following properties:

If an opponent observes M and CK(M), it should be computationally infeasible forthe opponent to construct a message M' such that CK(M') = CK(M)

CK(M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that CK(M) = CK(M') is 2^{-n} where n is the number of bits in the MAC.

Let M' be equal to some known transformation on M. i.e., M' = f(M).

MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES.

The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: D1, D2 … Dn. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code

(DAC) is calculated as follows: O1 = EK(D1)

O2 = EK(D2 O1) ⊕

O3 = EK(D3 O2) … ⊕

ON = EK(DN ON-1) ⊕

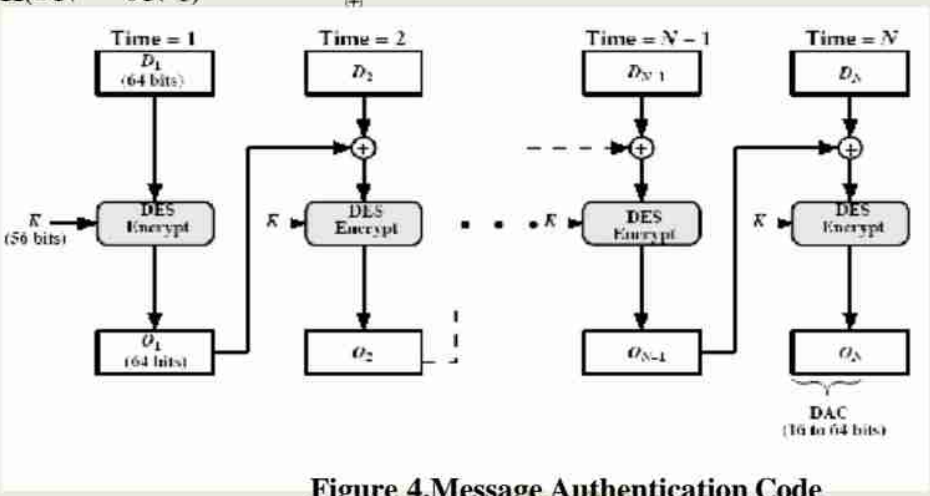


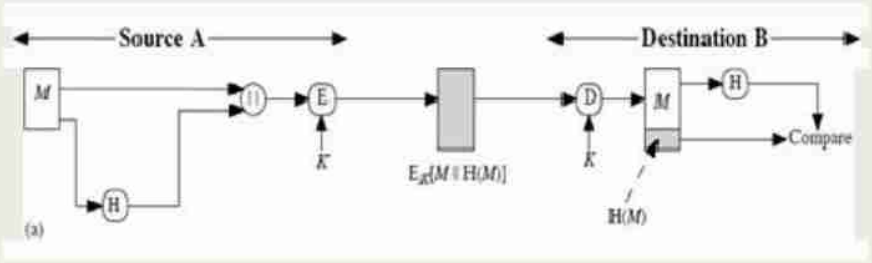
Figure 4.Message Authentication Code

IV. HASH FUNCTIONS

A variation on the message authentication code is the one-way hash function. As with MAC, a hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value.

There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

a) The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the 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.

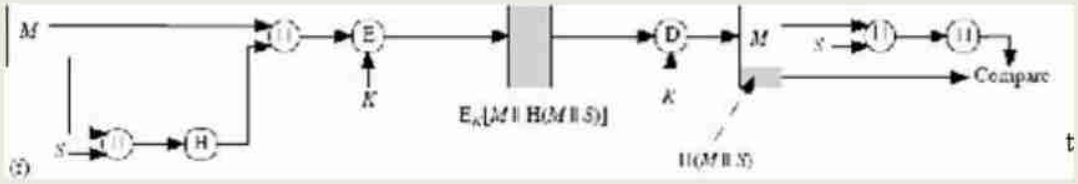


Figure 5. Basic uses of Hash function

c) Only the hash code is encrypted, using the public key encryption and using th sender's private key. It provides authentication plus the digital signature.

d) If confidentiality as well as digital signature is desired, then the message plus encrypted hash code can be encrypted using a symmetric secret key.

e) This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value 'S'. The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.

f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

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

Where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by re-computing the hash value.

Requirements for a Hash Function

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

H(y) = H(x). This is sometimes referred to as **weak collision resistance**.

5. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

The first three properties are requirements for the practical application of a hash function to message authentication. The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code. The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used.The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack, which we examine shortly.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can beexpressed as follows:

$C_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$

Where

C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

Thus, the probability that a data error will result in an unchanged hash value is 2^n. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2^{128}, the hash function on this type of data has an effectiveness of 2^{112}.

A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows:

1. Initially set the n-bit hash value to zero.
2. Process each successive n-bit block of data as follows:

a. Rotate the current hash value to the left by one bit. b. XOR the block into the hash value.

Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted

Message M, then an opponent would need to find an M' such that H(M') = H(M) to substitute another message and fool the receiver.

On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message

However, a different sort of attack is possible, based on **the birthday paradox** The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key

1. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. (Fraudulent message
2. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.
3. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

such as DES to compute the hash code G as follows:

Ho = initial value

Hi = EMi [Hi-1] G = HN

This is similar to the CBC technique, but in this case there is no secret key. As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.

Here is the scenario; we assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long:

1. Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code G.

2. Construct any desired message in the form Q1, Q2,..., QN2.

3. Compute for Hi = EQi [Hi-1]for $1 \leq i \leq (N-2)$.

4. Generate $2^{m/2}$ random blocks; for each block X, compute EX[HN-2.] Generate an additional $2^{m/2}$ random block; for each block Y, compute DY[G], where D is the decryption function corresponding to E.

5. Based on the birthday paradox, with high probability there will be an X and Y such that EX [HN-2] = DY[G].

6. Form the message Q1, Q2,..., QN-2, X, Y. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

This form of attack is known as a **meet-in-the-middle attack.**

Security of Hash Functions and Macs

Just as with symmetric and public-key encryption, we can group attacks on hash functions and MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

The nature of brute-force attacks differs somewhat for hash functions and MACs.

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm. Recall from our discussion of hash functions that there are three desirable properties:

One-way: For any given code h, it is computationally infeasible to find x such that H(x) = h.

Weak collision resistance: For any given block x, it is computationally infeasible to find y x with H(y) = H(x).

Strong collision resistance: It is computationally infeasible to find any pair (x, y) Such that H(x) = H(y).

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

One way	
Weak collision resistance	
Strong collision resistance	

Cryptanalysis

As with encryption algorithms, cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Hash Functions

In recent years, there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions. To understand these, we need to look at the overall structure of a typical secure hash function, and is the structure of most hash functions in use today, including SHA and Whirlpool.

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits.

The final block also includes the value of the total length of the input to the hash function. The inclusion of the length makes the job of the opponent more difficult.

Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.

The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-bit block) and produces an n-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, $b > n$; hence the term compression. The hash function can be summarized as follows:

$CV_0 = IV = \text{initial n-bit value}$ $CV_i = f(CV_{i-1}, Y_{i-1})$ $1 \leq i \leq L$ $H(M) = CV_L$

Where the input to the hash function is a message M consisting of the blocks $Y_0, Y_1, ..., Y_{L-1}$. The structure can be used to produce a secure hash function to operate on a message of any length.

Message Authentication Codes

There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs. Further, far less work has been done on developing such attacks.

Message Authentication.

i) Message authentication is concerned with:

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

ii) electronic equivalent of a signature on a message

iii) An **authenticator**, **signature**, or **message authentication code (MAC)** is sent along with the message

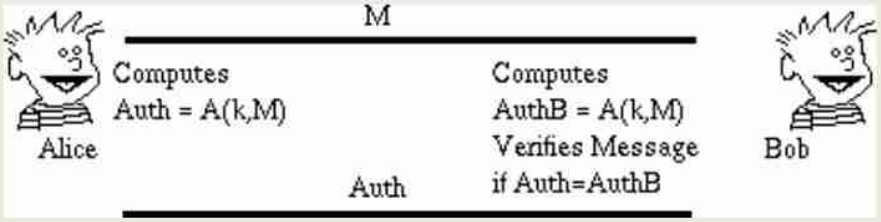
iv) The MAC is generated via some algorithm which depends on both the message and some (public or private) key known only to the sender and receiver

v) The message may be of any length

vi) the MAC may be of any length, but more often is some fixed size, requiring the use of some **hash function** to condense the message to the required size if this is not acheived by the authentication scheme

vii) need to consider replay problems with message and MAC

- require a message sequence number, timestamp or negotiated random values



Authentication using Private-key Ciphers

viii) if a message is being encrypted using a session key known only to the sender and receiver, then the message may also be authenticated

- since only sender or receiver could have created it

- any interference will corrupt the message (provided it includes sufficient redundancy to detect change)but this does not provide non-repudiation since it is impossible to prove who created the message

ix) message authentication may also be done using the standard modes of use of a block cipher

- sometimes do not want to send encrypted messages
- can use either CBC or CFB modes and send final block, since this will depend on all previous bits of the message
- no hash function is required, since this method accepts arbitrary length input and produces a fixed output
- usually use a fixed known IV
- this is the approached used in Australian EFT standards AS8205
- major disadvantage is small size of resulting MAC since 64-bits is probably too small

Hashing Functions

x) hashing functions are used to condense an arbitrary length message to a fixed size, usually for subsequent signature by a digital signature algorithm

xi) good cryptographic hash function h should have the following properties:

- h should destroy all holomorphic structures in the underlying public key cryptosystem (be unable to compute hash value of 2 messages combined given their individual hash values)
- h should be computed on the entire message
- h should be a one-way function so that messages are not disclosed by their signatures
- it should be computationally infeasible given a message and its hash value to compute another message with the same hash value
- should resist **birthday attacks** (finding any 2 messages with the same hash value, perhaps by iterating through minor permutations of 2 messages)

xii) it is usually assumed that the hash function is public and not keyed

xiii) traditional CRCs do not satisfy the above requirements

xiv) length should be large enough to resist birthday attacks (64-bits is now regarded as too small, 128-512 proposed)

MD2, MD4 and MD5

xv) family of one-way hash functions by Ronald Rivest

xvi) MD2 is the oldest, produces a 128-bit hash value, and is regarded as slower and less secure than MD4 and MD5

xvii) MD4 produces a 128-bit hash of the message, using bit operations on 32-bit operands for fast implementation

R L Rivest, "The MD4 Message Digest Algorithm", Advances in Cryptology - Crypto'90, Lecture Notes in Computer Science No 537, Springer-Verlag 1991, pp303-311

xviii) MD4 overview

- pad message so its length is 448 mod 512
- append a 64-bit message length value to message
- initialise the 4-word (128-bit) buffer (A,B,C,D)
- process the message in 16-word (512-bit) chunks, using 3 rounds of 16 bit operations each on the chunk & buffer
- output hash value is the final buffer value

xix) some progress at cryptanalysing MD4 has been made, with a small number of collisions having been found

xx) MD5 was designed as a strengthened version, using four rounds, a little more complex than in MD4 [2]

xxi) a little progress at cryptanalysing MD5 has been made with a small number of collisions having been found

xxii) both MD4 and MD5 are still in use and considered secure in most practical applications

xxiii) both are specified as Internet standards (MD4 in RFC1320, MD5 in RFC1321)

V. SHA (Secure Hash Algorithm)

xxiv) SHA was designed by NIST & NSA and is the US federal standard for use with the DSA signature scheme (nb the algorithm is SHA, the standard is SHS)

xxv) it produces 160-bit hash values

xxvi) SHA overview[3]

- pad message so its length is a multiple of 512 bits
- initialise the 5-word (160-bit) buffer (A,B,C,D,E) to
- (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
- process the message in 16-word (512-bit) chunks, using 4 rounds of 20 bit operations each on the chunk & buffer
- output hash value is the final buffer value

xxvii) SHA is a close relative of MD5, sharing much common design, but each having differences

xxviii) SHA has very recently been subject to modification following NIST identification of some concerns, the exact nature of which is not public

xxix) current version is regarded as secure

Digital Signature Schemes

xxx) public key signature schemes

xxxi) the private-key signs (creates) signatures, and the public-key verifies signatures

xxxii) only the owner (of the private-key) can create the digital signature, hence it can be used to verify who created a message

xxxiii) anyone knowing the public key can verify the signature (provided they are confident of the identity of the owner of the public key - the key distribution problem)

xxxiv) usually don't sign the whole message (doubling the size of information exchanged), but just a **hash** of the message

xxxv) digital signatures can provide non-repudiation of message origin, since an asymmetric algorithm is used in their creation, provided suitable timestamps and redundancies are incorporated in the signature

RSA

xxxvi) RSA encryption and decryption are commutative, hence it may be used directly as a digital signature scheme

- given an RSA scheme {(e,R), (d,p,q)}

xxxvii) to **sign** a message, compute:

- $S = M^d (\text{mod } R)$

xxxviii) to **verify** a signature, compute:

- $M = S^e (\text{mod } R) = M^{e.d} (\text{mod } R) = M (\text{mod } R)$

xxxix) thus know the message was signed by the owner of the public-key

xl) would seem obvious that a message may be encrypted, then signed using RSA without increasing it size

- but have blocking problem, since it is encrypted using the receivers modulus, but signed using the senders modulus (which may be smaller)
- several approaches possible to overcome this

xli) more commonly use a hash function to create a separate MDC which is then signed

El Gamal Signature Scheme

xlii) whilst the ElGamal encryption algorithm is not commutative, a closely related signature scheme exists

xliii) El Gamal Signature scheme

xliv) given prime p, public random number g, private (key) random number x, compute

- $y = g^x (\text{mod } p)$

xlv) public key is (y,g,p)

- nb (g,p) may be shared by many users
- p must be large enough so discrete log is hard

xlvi) private key is (x)

xlvii) to **sign** a message M

- choose a random number k, GCD(k,p-1)=1
- compute $a = g^k \pmod p$
- use extended Euclidean (inverse) algorithm to solve
- $M = x.a + k.b \pmod{p-1}$
- the signature is (a,b), k must be kept secret
- (like ElGamal encryption is double the message size)

xlviii) to **verify** a signature (a,b) confirm:

- $y^a.a^b \pmod p = g^M \pmod p$

Example of ElGamal Signature Scheme

xlix) given p=11, g=2

l) choose private key x=8

li) compute

- $y = g^x \pmod p = 2^8 \pmod{11} = 3$

lii) public key is y=3,g=2,p=11)

liii) to sign a message M=5

- choose random k=9
- confirm gcd(10,9)=1
- compute

$a = g^k \pmod p = 2^9 \pmod{11} = 6$

- solve

$M = x.a+k.b(\text{mod } p-1)$

$5 = 8.6+9.b(\text{mod } 10)$

giving $b = 3$

- signature is (a=6,b=3)

liv) to verify the signature, confirm the following are correct:

- $y^a.a^b(\text{mod } p) = g^M(\text{mod } p)$

o $3^6.6^3(\text{mod } 11) = 2^5(\text{mod } 11)$

DSA (Digital Signature Algorithm)

lv) DSA was designed by NIST & NSA and is the US federal standard signature scheme (used with SHA hash alg)

- DSA is the algorithm, DSS is the standard
- There was considerable reaction to its announcement!

debate over whether RSA should have been used

debate over the provision of a signature only alg

lvi) DSA is a variant on the ElGamal and Schnorr algorithms

lvii) description of DSA

- $p = 2^L$ a prime number, where L= 512 to 1024 bits and is a multiple of 64
- q a 160 bit prime factor of p-1
- $g = h^{(p-1)/q}$ where h is any number less than p-1 with $h^{(p-1)/q}(\text{mod } p) > 1$
- x a number less than q
- $y = g^x(\text{mod } p)$

lviii) to **sign** a message M

- generate random k, k<q
- compute
- $r = (g^k \pmod p) \pmod q$
- $s = k^{-1} \cdot SHA(M) + x.r \pmod q$
- the signature is (r,s)

lix) to **verify** a signature:

- $w = s^{-1} \pmod q$
- $u1 = (SHA(M).w) \pmod q$
- $u2 = r.w \pmod q$
- $v = (gu1.yu2 \pmod p) \pmod q$
- if v=r then the signature is verified

lx) comments on DSA

- was originally a suggestion to use a common modulus, this would make a tempting target, discouraged
- it is possible to do both ElGamal and RSA encryption using DSA routines, this was probably not intended :-)
- DSA is patented with royalty free use, but this patent has been contested, situation unclear
- Gus Simmons has found a subliminal channel in DSA, could be used to leak the private key from a library - make sure you trust your library implementer.

V. AUTHENTICATION SERVICES KERBEROS

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Kerberos relies exclusively on conventional encryption, making no use of public-key encryption.

The following are the requirements for Kerberos:

lxi) **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

lxii) **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.

lxiii) **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.

lxiv) **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

To support these requirements, the overall scheme of Kerberos is that of a trusted third- party authentication service that uses a protocol based on that proposed by Needham and Schroeder [NEED78] It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure.

A simple authentication dialogue

In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation. To counter this threat, servers must be able to confirm the identities of clients who request service. But in an open environment, this places a substantial burden on each server.

An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. The simple authentication dialogue is as follows:

A more secure authentication dialogue

There are two major problems associated with the previous approach:

Plaintext transmission of the password.

Each time a user has to enter the password.

To solve these problems, we introduce a scheme for avoiding plaintext passwords, and anew server, known as ticket granting server (TGS). The hypothetical scenario is as follows:

Once per user logon session:

1. C >> AS: IDc||IDtgs
2. AS >> C: Ekc (Tickettgs)

Once per type of service:

3. C >> TGS: IDc||IDv||Tickettgs
4. TGS >> C: ticketv

Once per service session:

5. C >> V: IDc||ticketv

Tickettgs= Ektgs(IDc||ADc||IDtgs||TS1||Lifetime1) Ticketv= Ekv(IDc||ADc||IDv||TS2||Lifetime2)

C: Client, AS: Authentication Server, V: Server, IDc : ID of the client, Pc:Password of the client, ADc: Address of client, IDv : ID of the server, Kv: secret key shared by AS and V, ||: concatenation, IDtgs: ID of the TGS server, TS1, TS2: time stamps, lifetime: lifetime of the ticket.

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

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service.

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password.

When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message.

If the correct password is supplied, the ticket is successfully recovered.Because only the correct usershould know the password, only the correct user can recover the ticket. Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext.

Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and4:

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.

4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

The service-granting ticket has the same structure as the ticket-granting ticket. Indeed, because the TGS is a server, we would expect that the same elements are needed to authenticate a client to the TGS and to authenticate a client to an application server. Again, the ticket contains a timestamp and lifetime. If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password. Note that the ticket is encrypted with a secret key (Kv) known only to the TGS and the server, preventing alteration.

Finally, with a particular service-granting ticket, the client can gain access to the corresponding service with step 5:

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID nd the service-granting ticket. The server authenticates by using the contents of the ticket.

This new scenario satisfies the two requirements of only one password query per user session and

protection of the user password.

Kerberos V4 Authentication Dialogue Message Exchange

Two additional problems remain in the more secure authentication dialogue:

Lifetime associated with the ticket granting ticket. If the lifetime is very short, then the user will be repeatedly asked for a password. If the lifetime is long, then the opponent has the greater opportunity for replay.

Requirement for the servers to authenticate themselves to users. The actual Kerberos protocol version 4 is as follows:

- a basic third-party authentication scheme
- have an Authentication Server (AS)
  – users initially negotiate with AS to identify self
  –AS provides a non-corruptible authentication credential (ticket granting ticket TGT)
- have a Ticket Granting server (TGS)
  –users subsequently request access to other services from TGS on basis of users

Message (1)	**Client requests ticket-granting ticket**
IDC	Tells AS identity of user from this client
ID_{tgs}	Tells AS that user requests access to TGS
TS1	Allows AS to verify that client's clock is synchronized with that of AS
Message (2)	**AS returns ticket-granting ticket**
Kc	Encryption is based on user's password, enabling AS and client to verify asswo d, and p t ctin contents of m ssa e (2)
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secureexchange between client and TGS without requiring them to share a
ID_{tgs}	Confirms that this ticket is for the TGS

The table given below illustrates the mode of dialogue in V4

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: $ID_c \parallel ID_{tgs} \parallel TS_1$
(2) AS → C: $E_{K_c}\left[K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$
(4) TGS → C: $E_{K_{c,tgs}}\left[K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2\right]$
$Ticket_v = E_{K_v}\left[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4\right]$
$Authenticator_c = E_{K_{tgs}}\left[ID_C \parallel AD_C \parallel TS_3\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: $Ticket_v \parallel Authenticator_c$
(6) V → C: $E_{K_{c,v}}\left[TS_5 + 1\right]$ (for mutual authentication)
$Ticket_v = E_{K_v}\left[K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4\right]$
$Authenticator_c = E_{K_{c,v}}\left[ID_C \parallel AD_C \parallel TS_5\right]$

TS2	Informs client of time this ticket was issued
Lifetime2	Informs client of the lifetime of this ticket
Tickettgs	Ticket to be used by client to access TGS
	(a) Authentication Service Exchange
Message (3)	**Client requests service-granting ticket**
IDV	Tells TGS that user requests access to server V
Tickettgs	Assures TGS that this user has been authenticated by AS
Authenticatorc	Generated by client to validate ticket
Message (4)	**TGS returns service-granting ticket**
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4)
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secureexchange between client and server without requiring them to share a
IDv	Confirms that this ticket is for server V
TS4	Informs client of time this ticket was issued
Ticketv	Ticket to be used by client to access server V
Tickettgs	Reusable so that user does not have to reenter password
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket
IDC	Indicates the rightful owner of this ticket
ADC	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_{tgs}	Assures server that it has decrypted ticket properly
TS2	Informs TGS of time this ticket was issued
Lifetime2	Prevents replay after ticket has expired
Authenticatorc	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay

$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering

IDc	Must match ID in ticket to authenticate ticket
ADc	Must match address in ticket to authenticate ticket
TS3	Informs TGS of time this authenticator was generated
	(b) Ticket-Granting Service Exchange
Message (5)	**Client requests service**
Ticketv	Assures server that this user has been authenticated by AS
Authenticatorc	Generated by client to validate ticket
Message (6)	Optional authentication of server to client
$K_{c,v}$	Assures C that this message is from V
TS5 + 1	Assures C that this is not a replay of an old reply
Ticketv	Reusable so that client does not need to request a new ticket from TGS for each access to the same server
Kv	Ticket is encrypted with key known only to TGS and server, to prevent tampering
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket
IDC	Indicates the rightful owner of this ticket
ADc	Prevents use of ticket from workstation other than one that initially requested the ticket
IDv	Assures server that it has decrypted ticket properly
TS4	Informs server of time this ticket was issued
Lifetime4	Prevents replay after ticket has expired
Authenticatorc	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, to

	prevent tampering
IDC	Must match ID in ticket to authenticate ticket
ADc	Must match address in ticket to authenticate ticket
TS5	Informs server of time this authenticator was generated
	(c) Client/Server Authentication

VI. Kerberos 4 Overview

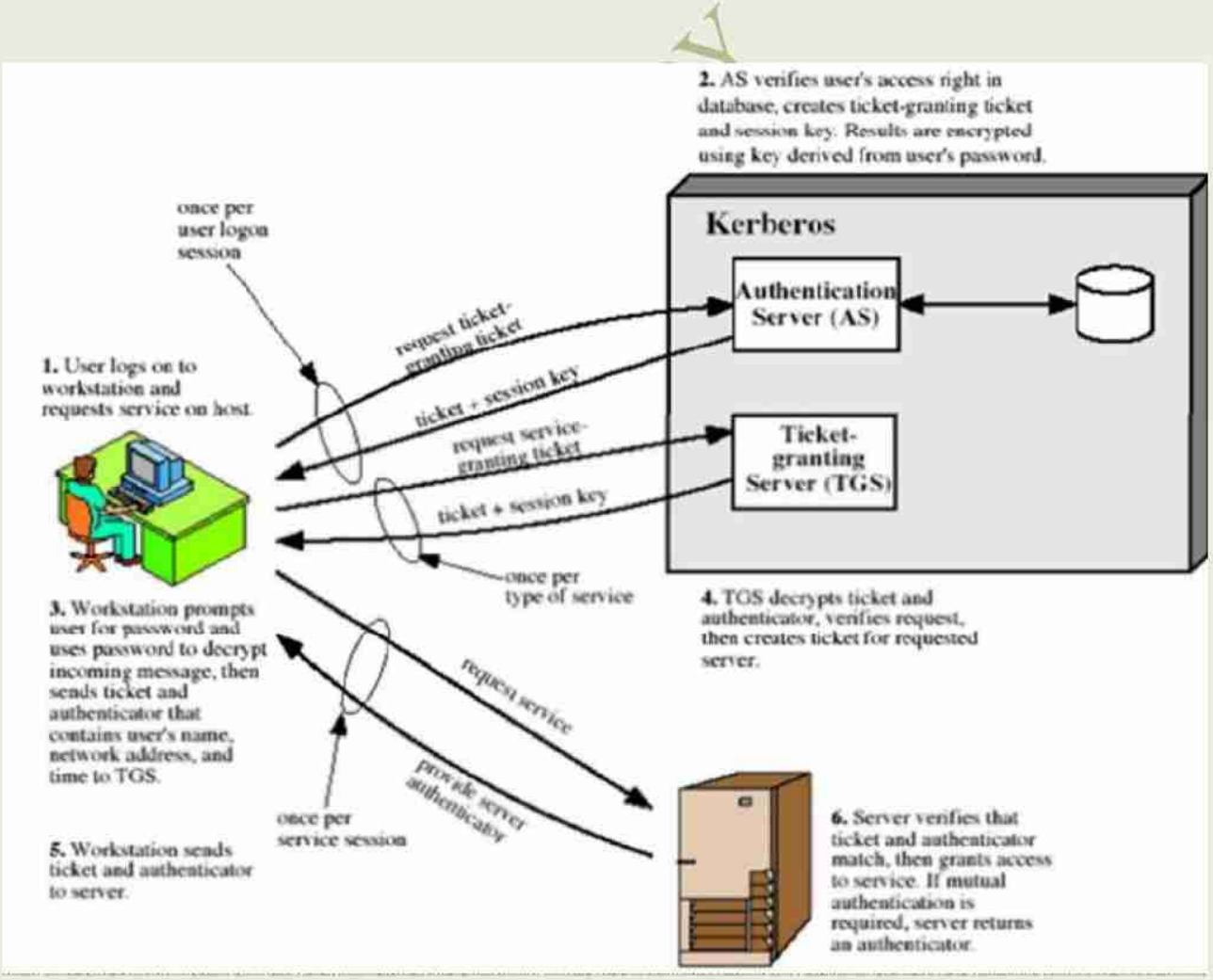


Figure 6. Kerberos 4

Kerberos Realms and Multiple Kerberi

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

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

2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.Such an environment is referred to as a **Kerberos realm**.

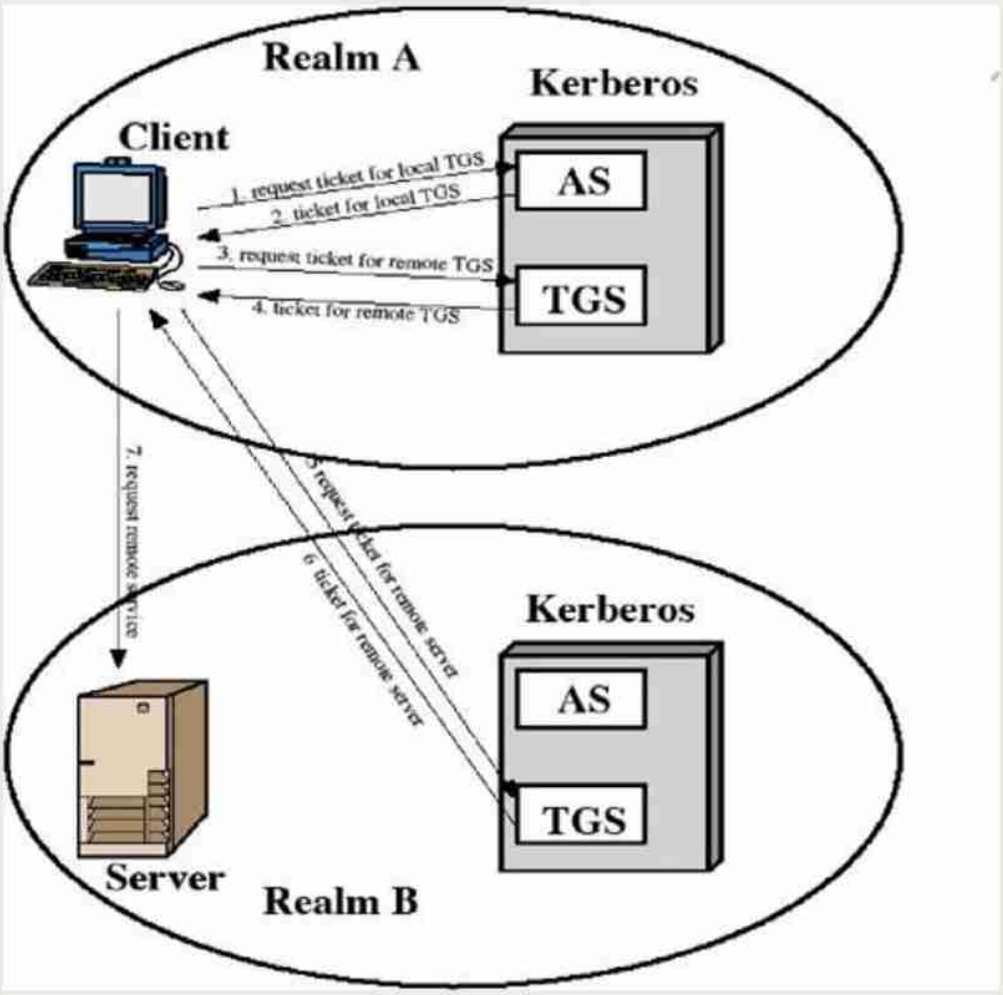


Figure 7. Kerberos realm

The concept of *realm* can be explained as follows.

A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room.

A read-only copy of the Kerberos database might also reside on other Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system.

Each Kerberos principal is identified by its principal name. Principal names consist of three parts: aservice or user name, an instance name, and a realm name

Networks of clients and servers under different administrative organizations typically constitute different realms.

That is, it generally is not practical, or does not conform to administrative policy, to have users andservers in one administrative domain registered with a Kerberos server elsewhere.

However, users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

Kerberos provides a mechanism for supporting such inter realm authentication. For two realms to support inter realm authentication, a third requirement is added:

3. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users. Furthermore, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

Kerberos version 5

Version 5 of Kerberos provides a number of improvements over version 4.

- developed in mid-1990's
- provides improvements over v4
  - addresses environmental shortcomings and technical deficiencies
- specified as Internet standard RFC 1510

Differences between version 4 and 5

Version 5 is intended to address the limitations of version 4 in two areas:

Environmental shortcomings

- o encryption system dependence
- o internet protocol dependence
- o message byte ordering
- o ticket lifetime
- o authentication forwarding
- o inter-realm authentication

Technical deficiencies

- o double encryption o PCBC encryption o Session keys
- o Password attacks

The version 5 authentication dialogue

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: Options $\|$ ID_c $\|$ $Realm_c$ $\|$ ID_{tgs} $\|$ $Times$ $\|$ $Nonce_1$
(2) AS → C: $Realm_c \| ID_C \| Ticket_{tgs} \| E_{K_c}\left[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: Options $\|$ ID_v $\|$ $Times$ $\|$ $\|$ $Nonce_2$ $\|$ $Ticket_{tgs}$ $\|$ $Authenticator_c$
(4) TGS → C: $Realm_c \| ID_C \| Ticket_v \| E_{K_{c,tgs}}\left[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_V\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| Realm_c \| TS_1\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: Options $\|$ Ticket_v $\|$ Authenticator_c
(6) V → C: $E_{K_{c,V}}$ [TS_2 $\|$ Subkey $\|$ Seq#]
$Ticket_v = E_{K_v}\left[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times\right]$
$Authenticator_c = E_{K_{c,V}}\left[ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#\right]$

The following new elements are added:

Realm: Indicates realm of user
Options: Used to request that certain flags be set in the returned ticket Times:
Used by the client to request the following time settings in the ticket:

from: the desired start time for the requested ticket

till: the requested expiration time for the requested ticket rtime: requested renew-till time

Nonce: A random value to be repeated in message (2) to assure that the response is fresh and has not been replayed by an opponent

Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password.
This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information.

The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options.

These flags introduce significant new functionality to version 5. For now, we defer a discussion of these flags and concentrate on the overall structure of the version 5 protocol.

Let us now compare the ticket-granting service exchange for versions 4 and 5. We see that message (3) for both versions includes an authenticator, a ticket, and the name of the requested service.

In addition, version 5 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1).

The authenticator itself is essentially the same as the one used in version 4.

Message (4) has the same structure as message (2), returning a ticket plus information needed by

the client, the latter encrypted with the session key now shared by the client and the TGS.

Finally, for the client/server authentication exchange, several new features appear in version 5. In message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields as follows:

Subkey: The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket (Kc,v) is used.

Sequence number: An optional field that specifies the starting sequence number to be use may be sequence numbered to detect replays.

If mutual authentication is required, the server responds with message (6). This message includes the timestamp from the authenticator. Note that in version 4, the timestamp was incremented by one. This is not necessary in version 5 because the nature of the format of messages is such that it is not possible for an opponent to create message (6) without knowledge of the appropriate encryption keys.

Ticket Flags

The flags field included in tickets in version 5 supports expanded functionality compared to that available in version 4.

X.509 Certificates

Overview:

- **issued by a Certification Authority (CA), containing:**

– version (1, 2, or 3)

– serial number (unique within CA) identifying certificate

– signature algorithm identifier

- issuer X.500 name (CA)
- period of validity (from - to dates)
- subject X.500 name (name of owner)
- subject public-key info (algorithm, parameters, key)
- issuer unique identifier (v2+)
- subject unique identifier (v2+)
- extension fields (v3)
- signature (of hash of all fields in certificate)

- **notation CA<<A>> denotes certificate for A signed by CA**

  X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

  X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts. For example, the X.509 certificate format is used in S/MIME), IP Security and SSL/TLS and SET

  X.509 is based on the use of public-key cryptography and digital signatures. The standard does not

dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function.

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

Version:

Differentiates among successive versions of the certificate format; the default is version 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If oneor more extensions are present, the version must be version 3.

Serial number:

An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.

Signature algorithm identifier:

The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility.

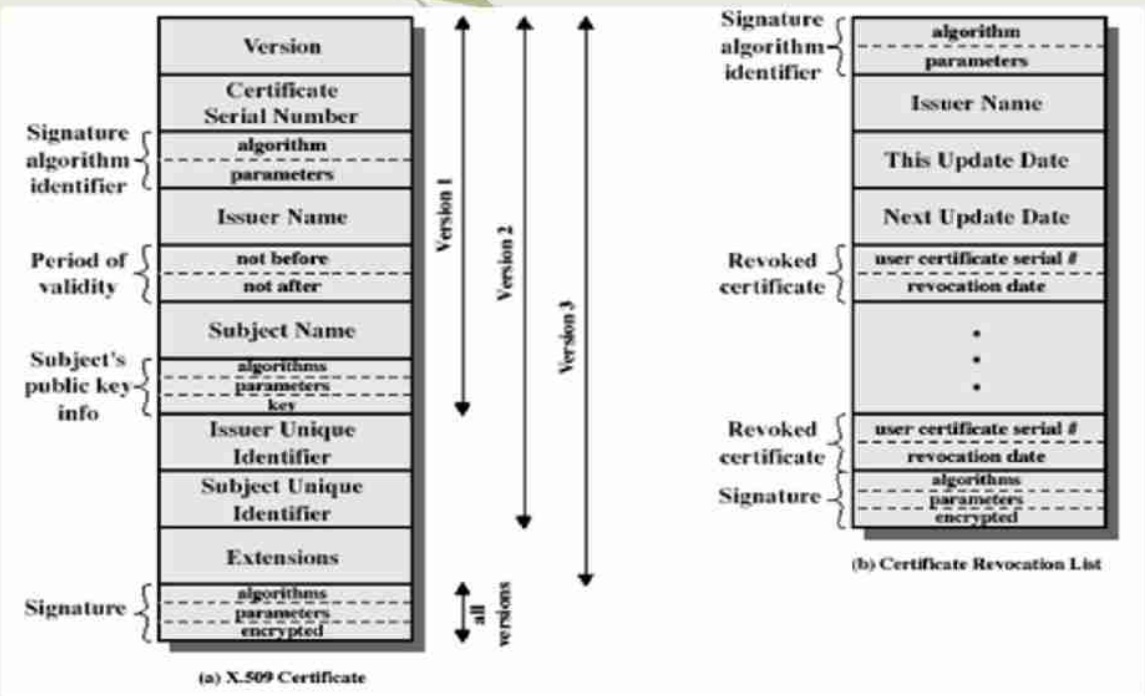


Figure 8. X.509 C

Issuername:

X.500 name of the CA that created and signed this certificate.

Period of validity:

Consists of two dates: the first and last on which the certificate is valid.

Subject name:

The name of the user to whom this certificate refers. That is, this certificate certifies the public keyof the subject who holds the corresponding private key.

Subject's public-key information:

The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

Issuerunique identifier:

An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

Subject unique identifier:

An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

Extensions:

A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.

Signature:

Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifie

The standard uses the following notation to define a certificate: CA<<A>> = CA {V, SN, AI, CA, TA, A, Ap} where

Y <<X>> = the certificate of user X issued by certification authority Y Y {I} = the signing of I by Y. It consists of I with an encrypted hash code appended

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

Obtaininga User's Certificate

User certificates generated by a CA have the following characteristics:

Any user with access to the public key of the CA can verify the user public key that was certified.

No party other than the certification authority can modify the certificate without this being detected.

ecause certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

If all users subscribe to the same CA, then there is a common trust of that CA. All user certificates can be placed in the directory for access by all users.

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each participating user must have a copy of the CA's own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

Now suppose that A has obtained a certificate from certification authority X1 and B has obtained a certificate from CA X2. If A does not securely know the public key of X2, then B's certificate, issued by X2, is useless to A.

A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:

1. A obtains, from the directory, the certificate of X2 signed by X1. Because A securely knows X1's public key, A can obtain X2's public key from its certificate and verify it by means of X1's signature on the certificate.
2. A then goes back to the directory and obtains the certificate of B signed by X2

Because A now has a trusted copy of X2's public key, A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

X1<<X2>> X2 <<B>>

In the same fashion, B can obtain A's public key with the reverse chain: X2<<X1>> X1 <<A>>This scheme need not be limited to a chain of two certificates. An arbitrarily long path of
CAs can be followed to produce a chain. A chain with N elements would be expressed as

X1<<X2>> X2 <<X3>>... XN<<B>>

In this case, each pair of CAs in the chain (Xi, Xi+1) must have created certificates for each other.

All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public-key certificate. X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.

Forward certificates: Certificates of X generated by other CAs
Reverse certificates: Certificates generated by X that are the certificates of other CAs

CA Hierarchy Use

In the example given below , user A can acquire the following certificates from the directory to establish a certification path to B:

X<<W>> W <<V>> V <<Y>> <<Z>> Z <<B>>

When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key. Using this public key, A can send encrypted

Messages to B. If A wishes to receive encrypted messages back from B, or to sign messages sent to B, then B will require A's public key, which can be obtained from the following certification path:

Z<<Y>> Y <<V>> V <<W>> W <<X>>X <<A>>

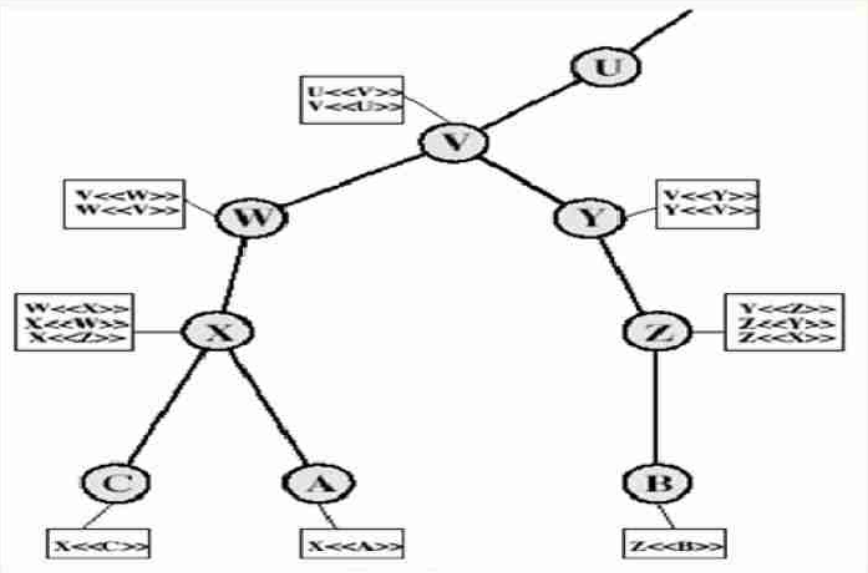


Figure 9.X.509 hierarchy

B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.

Certificate Revocation

- Certificates have a period of validity
- may need to revoke before expiry, for the following reasons eg:

1. user's private key is compromised
2. User is no longer certified by this CA
3. CA's certificate is compromised

- CA's maintain list of revoked certificates
  1. the Certificate Revocation List (CRL)
- users should check certs with CA's CRL

Authentication Procedures

X.509 includes three alternative authentication procedures:

- **One-Way Authentication**
- **Two-Way Authentication**
- **Three-Way Authentication**
- all use public-key signatures

One-Way Authentication

- 1 message (A->B) used to establish
  - the identity of A and that message is from A
  - message was intended for B
  - integrity & originality of message
- message must include timestamp, nonce, B's identity and is signed by A

Two-Way Authentication

- 2 messages (A->B, B->A) which also establishes in addition:
  - the identity of B and that reply is from B
  - that reply is intended for A
  - integrity & originality of reply

reply includes original nonce from A, also timestamp and nonce from B

Three-Way Authentication

- 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks
- has reply from a back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

X.509 Version 3

The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. [FORD95] lists the following requirements not satisfied by version 2:

1. The Subject field is inadequate to convey the identity of a key owner to a public- key user.

2. The Subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other Internet- related identification.

3. There is a need to indicate security policy information. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.

4. It is important to be able to identify different keys used by the same owner at different times.

The certificate extensions fall into three main categories: key and policy information, subject and issuer attributes, and certification path constraints.

Key and Policy Information

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy.. For example, a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes the following:

Authority key identifier: Identifies the public key to be used to verify the signature on this certificate or CRL.

Subject key identifier: Identifies the public key being certified. Useful for subject key pair updating.

Key usage: Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used.

Private-key usage period: Indicates the period of use of the private key corresponding to the public key.. For example, with digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key.

Certificate policies: Certificates may be used in environments where multiple policies apply.

Policy mappings: Used only in certificates for CAs issued by other CAs.

Certificate Subject and Issuer Attributes

These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject, to increase a certificate user's confidence that the certificate subject is a particular person or entity. For example, information such as postal address, position within a corporation, or picture image may be required.

The extension fields in this area include the following:

Subject alternative name: Contains one or more alternative names, using any of a variety of forms

Subject directory attributes: Conveys any desired X.500 directory attribute values for the subject of this certificate.

Certification Path Constraints

These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs.The extension fields in this area include the following:

Basic constraints: Indicates if the subject may act as a CA. If so, a certification

pathlength constraint may be specified.

Name constraints: Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.

Policy constraints: Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

UNIT 5 NETWORKSECURITY AND SYSTEM SECURITY

Syllabus:

Electronic Mail Security – IP Security – Web Security – Intruders – Malicious S/Ws – Firewalls

5.1 Electronic Mail Security

Overview

- Pretty Good Privacy (PGP)
- S/MIME
- DomainKeys Identified Mail (DKIM)

5.2 Email Security Enhancements

- Confidentiality: Protection from disclosure
- Authentication: Of sender of message
- Message integrity: Protection from modification
- Non-repudiation of origin: Protection from denial by sender

5.3 PGP – Authentication and Confidentiality

2013, when the *NSA (United States National Security Agency) scandal* was leaked to the public, people started to opt for the services which can provide them a strong privacy for theirdata. Among the services people opted for, most particularly for Emails, were different plug- ins and extensions for their browsers. Interestingly, among the various plug-ins and extensions that people started to use, there were two main programs that were solely responsible for the complete email security that the people needed. One was S/MIME which we will see later and the other was PGP.

As said, PGP (Pretty Good Privacy), is a popular program that is used to provide confidentiality and authentication services for electronic mail and file storage. It was designed by Phil Zimmermann way back in 1991. He designed it in such a way, that the best cryptographic algorithms such as RSA, Diffie-Hellman key exchange, DSS are used for the public-key encryption (or) asymmetric encryption; CAST-128, 3DES, IDEA are used for symmetric encryption and SHA-1 is used for hashing purposes. PGP software is an open source one and is not dependent on either of the OS (Operating System) or the processor. Theapplication is based on a few commands which are very easy to use.

The following are the services offered by PGP:

- Authentication
- Confidentiality
- Compression
- Email Compatibility
- Segmentation

Authentication:

Authentication basically means something that is used to validate something as true or real.To login into some sites sometimes we give our account name and password that is an authentication verification procedure.

In the email world, checking the authenticity of an email is nothing but to check *whether it actually came from the person it says*. In emails, authentication has to be checked as there aresome people who spoof the emails or some spams and sometimes it can cause a lot of inconvenience. The Authentication service in PGP is provided as follows:

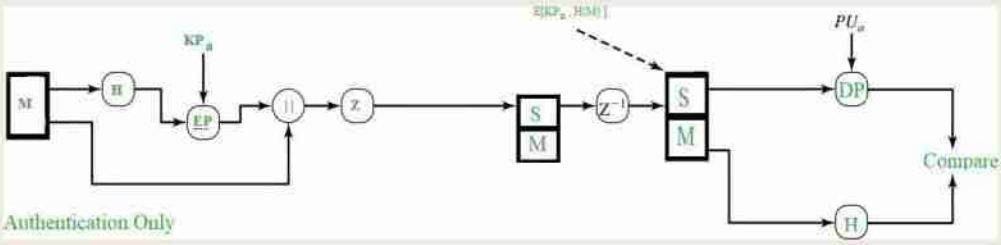


Figure 5.1 Authentication

As shown in the above figure, the Hash Function (H) calculates the Hash Value of the message. For the hashing purpose, SHA-1 is used and it produces a 160-bit output hash value.Then, using the sender's private key (KP_a), it is encrypted and it's called as Digital Signature.The Message is then appended to the signature. All the process happened till now, is sometimes described as *signing the message.* Then the message is compressed to reduce the transmission overhead and is sent over to the receiver.

At the receiver's end, the data is decompressed and the message, signature are obtained. The signature is then decrypted using the sender's public key (PU_a) and the hash value is obtained. The message is again passed to hash function and it's hash value is calculated and obtained.

Both the values, one from signature and another from the recent output of hash function are compared and if both are same, it means that the email is actually sent from a known one and is legit, else it means that it's not a legit one.

Confidentiality:

Sometimes we see some packages labelled as 'Confidential', which means that those packages are not meant for all the people and only selected persons can see them. The same applies to the email confidentiality as well. Here, in the email service, only the sender and thereceiver should be able to read the message, that means the contents have to be kept secret from every other person, except for those two.

PGP provides that Confidentiality service in the following manner:

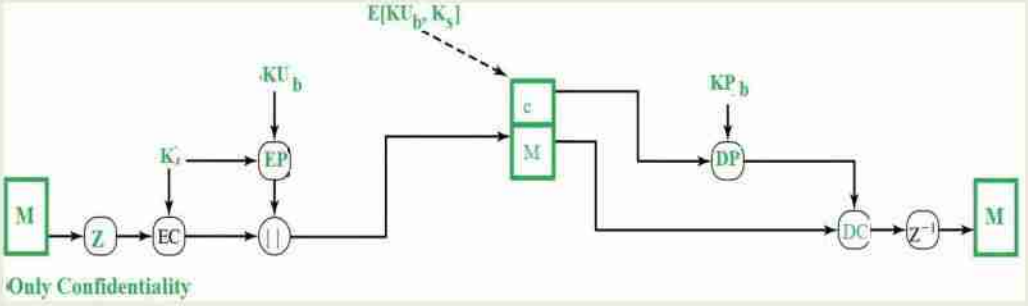


Figure 5.2 Confidentiality

The message is first compressed and a 128-bit session key (K_s), generated by the PGP, isused to encrypt the message through symmetric encryption. Then, the session key (K_s) itself gets encrypted through public key encryption (EP) using receiver's public key(KU_b) . Boththe encrypted entities are now concatenated and sent to the receiver.

As you can see, the original message was compressed and then encrypted initially and hence even if anyone could get hold of the traffic, he cannot read the contents as they are not in readable form and they can only read them if they had the session key (K_s). Even though session key is transmitted to the receiver and hence, is in the traffic, it is in encrypted form and only the receiver's private key (KP_b)can be used to decrypt that and thus our message would be completely safe.

At the receiver's end, the encrypted session key is decrypted using receiver's private key (KP_b) and the message is decrypted with the obtained session key. Then, the message is decompressed to obtain the original message (M).

RSA algorithm is used for the public-key encryption and for the symmetric jey encryption, CAST-128(or IDEA or 3DES) is used.

Practically, both the Authentication and Confidentiality services are provided in parallel asfollows:

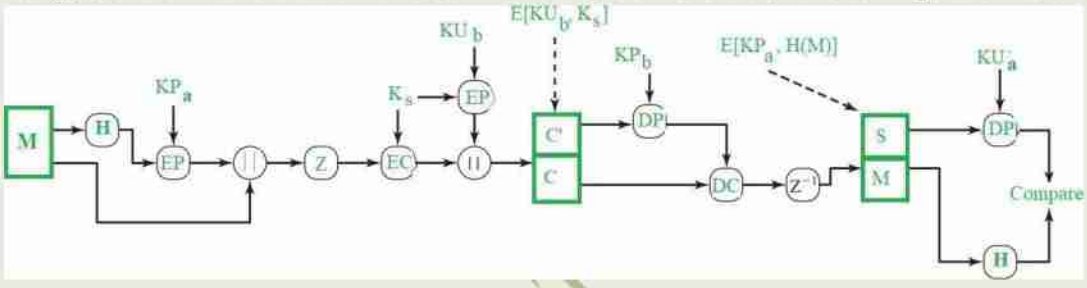


Figure 5.3 Authentication and Confidentiality

Note:
M – Message
H – Hash Function
K_s – A random Session Key created for Symmetric Encryption purposeDP – Public-Key Decryption Algorithm
EP – Public-Key Encryption Algorithm DC – Asymmetric Encryption AlgorithmEC – Symmetric Encryption Algorithm
KP_b – A private key of user B used in Public-key encryption processKP_a – A private key of user A used in Public-key encryption processPU_a – A public key of user A used in Public-key encryption process PU_b – A public key of user B used in Public-key encryption process
|| – Concatenation
Z – Compression Function
Z^{-1} – Decompression Function

5.4. Secure /Multipurpose Internet Mail Extensions (S/MIME)

S/MIME

S/MIME is a protocol for the secure exchange of e-mail and attached documents originally developed by RSA Security. Secure/Multipurpose Internet Mail Extensions (S/MIME) adds security to Internet e-mail based on the Simple Mail Transfer Protocol (SMTP) method and adds support for digital signatures and encryption to SMTP mail to support authentication of the sender and privacy of the communication.

Note that because HTTP messages can transport MIME data, they can also use S/MIME.

Working of S/MIME

S/MIME is an extension of the widely implemented Multipurpose Internet Mail Extensions (MIME) encoding standard, which defines how the body portion of an SMTP message is structured and formatted. S/MIME uses the RSA public key cryptography algorithm along with the Data Encryption Standard (DES) or Rivest-Shamir-Adleman (RSA) encryptionalgorithm. In an S/MIME message, the MIME body section consists of a message in PKCS#7 format that contains an encrypted form of the MIME body parts. The MIME content type for the encrypted data is application/pkcs7-mime.

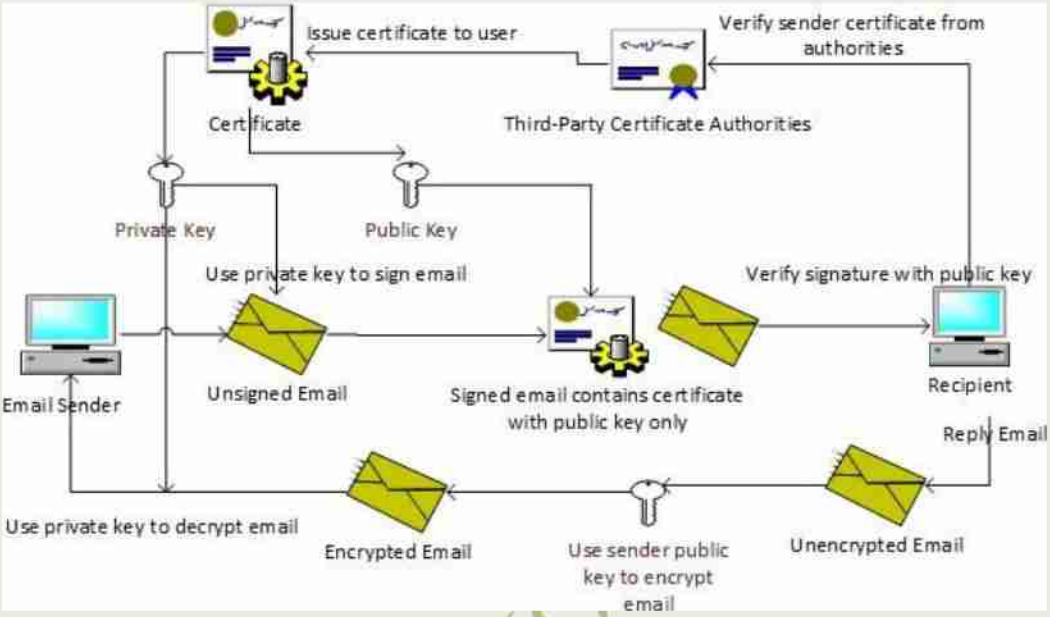


Figure 5.4 S/Mime Structure

Understanding Digital Signatures

Digital signatures are the more commonly used service of S/MIME. As the name suggests, digital signatures are the digital counterpart to the traditional, legal signature on a paper document. As with a legal signature, digital signatures provide the following security capabilities:

- Authentication A signature serves to validate an identity. It verifies the answer to "who are you" by providing a means of differentiating that entity from all others and proving its uniqueness. Because there is no authentication in SMTP e-mail, there is no way to know who actually sent a message. Authentication in a digital signature solves this problem by allowing a recipient to know that a message was sent by the person or organization who claims to have sent the message.

- Nonrepudiation The uniqueness of a signature prevents the owner of the signature from disowning the signature. This capability is called nonrepudiation. Thus, the authentication that a signature provides gives the means to enforce nonrepudiation. The concept of nonrepudiation is most familiar in the context of paper contracts: a signed contract is a legally binding document, and it is impossible to disown an authenticated signature. Digital signatures provide the same function and, increasingly in some areas, are recognized as legally binding, similar to a signature on paper. Because SMTP e-mail does not provide a means of authentication, it

cannot provide nonrepudiation. It is easy for a sender to disavow ownership of an SMTP e-mail message.

- Data integrity An additional security service that digital signatures provide is data integrity. Data integrity is a result of the specific operations that make digital signatures possible. With data integrity services, when the recipient of a digitally signed e-mail message

validates the digital signature, the recipient is assured that the e-mail message that is received is, in fact, the same message that was signed and sent, and has not been altered while in transit. Any alteration of the message while in transit after it has been signed invalidates the signature. In this way, digital signatures are able to provide an assurance that signatures on paper cannot, because it is possible for a paper document to be altered after it has been signed.

5.5 Secure Electronic Transaction (SET) Protocol

Secure Electronic Transaction or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT) and NetScape which provided technology of Secure Socket Layer (SSL).

SET protocol restricts revealing of credit card details to merchants thus keeping hackers and thieves at bay. SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

Before discussing SET further, let's see a general scenario of electronic transaction, whichincludes client, payment gateway, client financial institution, merchant and merchant financial institution.

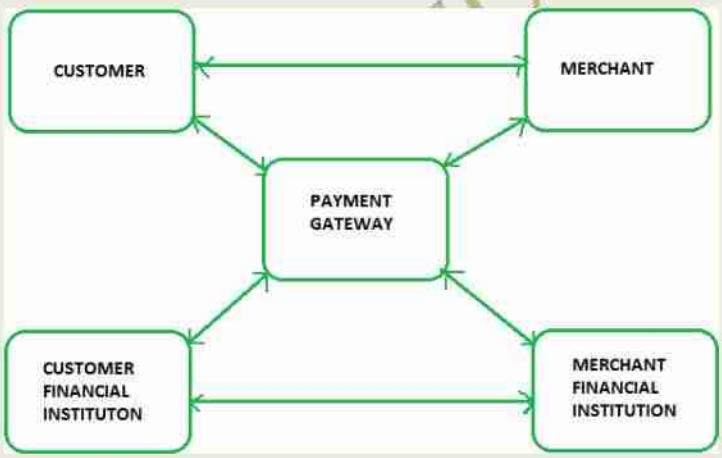


Figure 5.5 SET protocol

Requirements in SET:
SET protocol has some requirements to meet, some of the important requirements are :

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.

- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of best security mechanisms.

Participants in SET :
In the general scenario of online transaction, SET includes similar participants:

- Cardholder – customer
- Issuer – customer financial institution
- Merchant
- Acquirer – Merchant financial
- Certificate authority – Authority which follows certain standards and issues certificates (like X.509V3) to all other participants.

SET functionalities:

- Provide Authentication
  - Merchant Authentication – To prevent theft, SET allows customers to check previous relationships between merchant and financial institution. Standard X.509V3 certificates are used for this verification.
  - Customer / Cardholder Authentication – SET checks if use of credit card is done by an authorized user or not using X.509V3 certificates.
- Provide Message Confidentiality: Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purpose.
- Provide Message Integrity: SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1,

Dual Signature:

The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers:

- Order Information (OI) for merchant
- Payment Information (PI) for bank.

You might think sending them separately is an easy and more secure way, but sending themin a connected form resolves any future dispute possible. Here is the generation of dual signature:

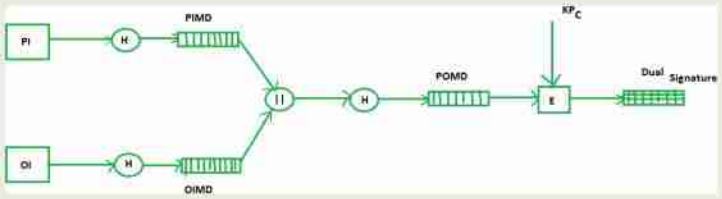


Figure 5.6 Generation of dual signature

Where,

PI stands for payment informationOI
stands for order information
PIMD stands for Payment Information Message Digest
OIMD stands for Order Information Message Digest
POMD stands for Payment Order Message Digest
H stands for Hashing
E stands for public key encryption
KPc is customer's private key
|| stands for append operation
Dual signature, DS= E(KPc, [H(H(PI)||H(OI))])

Purchase Request Generation:
The process of purchase request generation requires three inputs:

- Payment Information (PI)
- Dual Signature
- Order Information Message Digest (OIMD)

The purchase request is generated as follows:

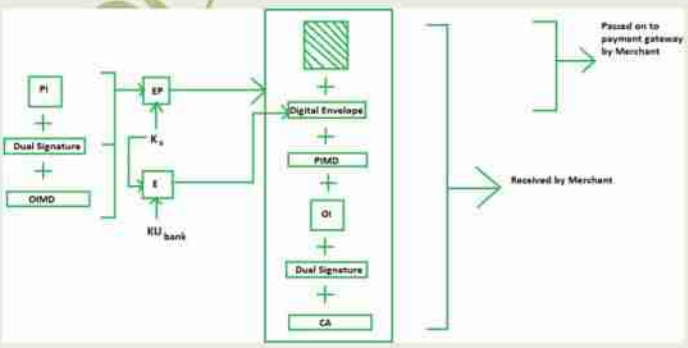


Figure 5.7 Purchase Request Generation

Here,

PI, OIMD, OI all have the same meanings as before.The

new things are :

EP which is symmetric key encryptionKs

is a temporary symmetric key KUbank is

public key of bank

CA is Cardholder or customer Certificate

Digital Envelope = E(KUbank, Ks)

Purchase Request Validation on Merchant Side :
The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows:

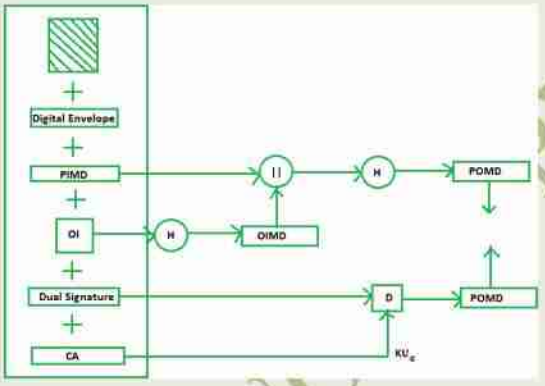


Figure 5.8 POMD generated through decryption of Dual Signature

Since we used Customer private key in encryption here we use KUc which is public key of customer or cardholder for decryption 'D'.

Payment Authorization and Payment Capture:

Payment authorization as the name suggests is the authorization of payment information by merchant which ensures payment will be received by merchant. Payment capture is the process by which merchant receives payment which includes again generating some request blocks to gateway and payment gateway in turn issues payment to merchant.

5.6 IP security (IPSec)

The IP sec4rity (IPSec) is an Internet Engineering Task Force (IETF) standard suite of protocols between 2 communication points across the IP network that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets. The protocols needed for secure key exchange and key management are defined in it.

Uses of IP Security –
IPsec can be used to do the following things:

- To encrypt application layer data.
- To provide security for routers sending routing data across the public internet.
- To provide authentication without encryption, like to authenticate that the data originates from a known sender.
- To protect network data by setting up circuits using IPsec tunneling in which all data is being sent between the two endpoints is encrypted, as with a Virtual Private Network(VPN) connection.

Components of IP Security –
It has the following components:

- Encapsulating Security Payload (ESP): It provides data integrity, encryption, authentication and anti-replay. It also provides authentication for payload.
- Authentication Header (AH) : It also provides data integrity, authentication and anti-replay and it does not provide encryption. The anti-replay protection, protects against unauthorized transmission of packets. It does not protect data's confidentiality.

Figure 5.9 Authentication Header

- Internet Key Exchange (IKE)

It is a network security protocol designed to dynamically exchange encryption keys and finda way over Security Association (SA) between 2 devices. The Security Association (SA) establishes shared security attributes between 2 network entities to support securecommunication. The Key Management Protocol (ISAKMP) and Internet Security Associationwhich provides a framework for authentication and key exchange. ISAKMP tells how the setup of the Security Associations (SAs) and how direct connections between two hosts that are using IPsec.

Internet Key Exchange (IKE) provides message content protection and also an open frame for implementing standard algorithms such as SHA and MD5. The algorithm's IP sec users produces a unique identifier for each packet. This identifier then allows a device to determinewhether a packet has been correct or not. Packets which are not authorized are discarded and not given to receiver.

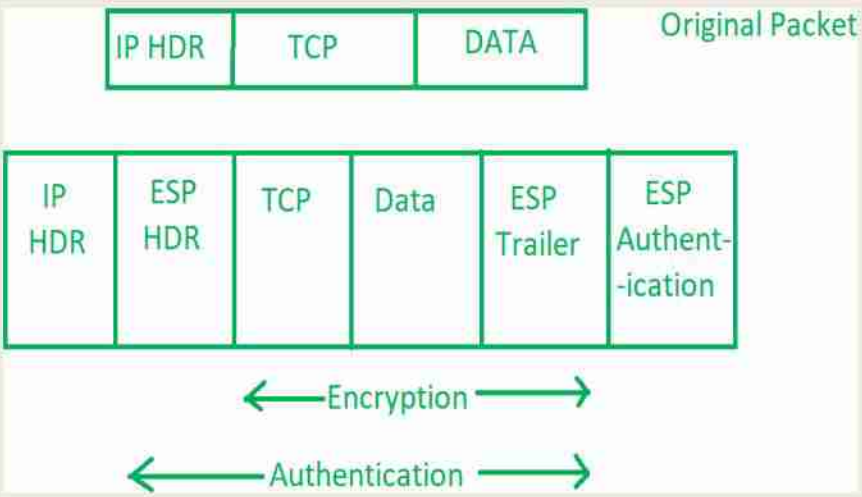


Figure 5.10 Internet Key Exchange

Working of IP Security –

- The host checks if the packet should be transmitted using IPsec or not. These packet traffic triggers the security policy for themselves. This is done when the system sending the packet apply an appropriate encryption. The incoming packets are also checked by the host that they are encrypted properly or not.
- Then the IKE Phase 1 starts in which the 2 hosts (using IPsec) authenticate themselves to each other to start a secure channel. It has 2 modes. The Main mode which provides the greater security and the Aggressive mode which enables the host to establish an IPsec circuit more quickly.
- The channel created in the last step is then used to securely negotiate the way the IP circuit will encrypt data accross the IP circuit.
- Now, the IKE Phase 2 is conducted over the secure channel in which the two hosts negotiate the type of cryptographic algorithms to use on the session and agreeing on secret keying material to be used with those algorithms.
- Then the data is exchanged across the newly created IPsec encrypted tunnel. These packets are encrypted and decrypted by the hosts using IPsec SAs.
- When the communication between the hosts is completed or the session times out then the IPsec tunnel is terminated by discarding the keys by both the hosts.

IPSec Architecture

IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture include protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integrity

IP Security Architecture:

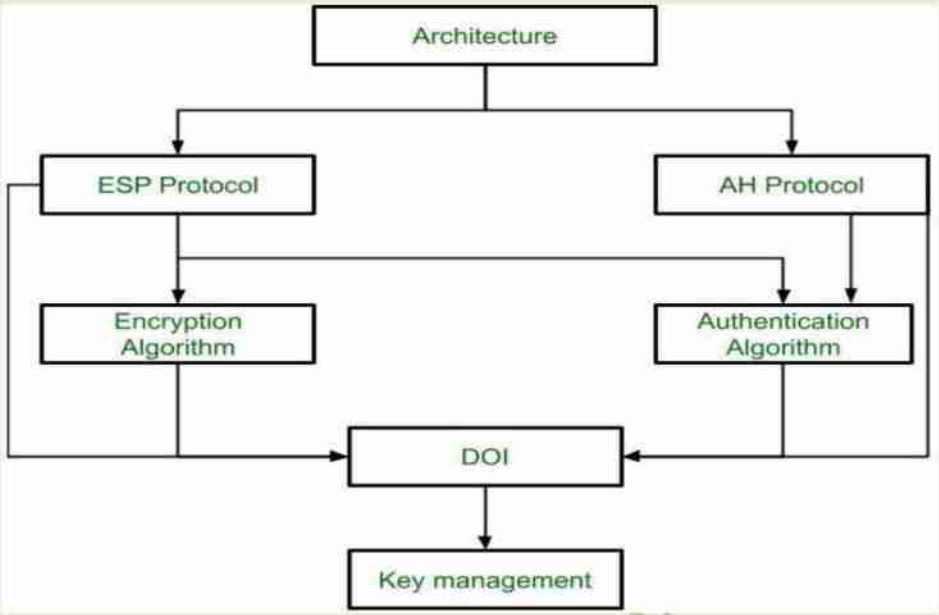


Fig 5.11 IP Security Architecture

- **Architecture:**
  Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.
- **ESP Protocol:**
  ESP (Encapsulation Security Payload) provide the confidentiality service. Encapsulation Security Payload is implemented in either two ways:
  - ESP with optional Authentication.
  - ESP with Authentication.

Packet Format:

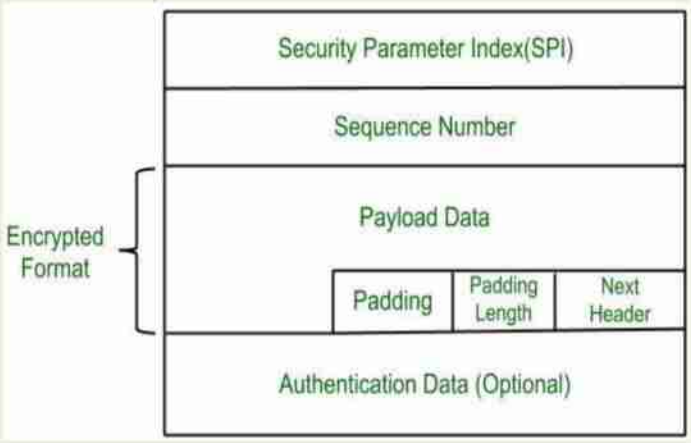


Fig 5.12 Packet Format

- **Security Parameter Index (SPI):**
  This parameter is used in Security Association. It is used to give a unique number tothe connection build between Client and Server.
- **Sequence Number:**
  Unique Sequence number are allotted to every packet so that at the receiver sidepackets can be arranged properly.
- **Payload Data:**
  Payload data means the actual data or the actual message. The Payload data is in encrypted format to achieve confidentiality.
- **Padding:**
  Extra bits or space added to the original message in order to ensure confidentiality. Padding length is the size of the added bits or space in the original message.
- **Next Header:**
  Next header means the next payload or next actual data.
- **Authentication Data**
  This field is optional in ESP protocol packet format.

- Encryption algorithm:
  Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.
- **AH Protocol:**
  AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity.

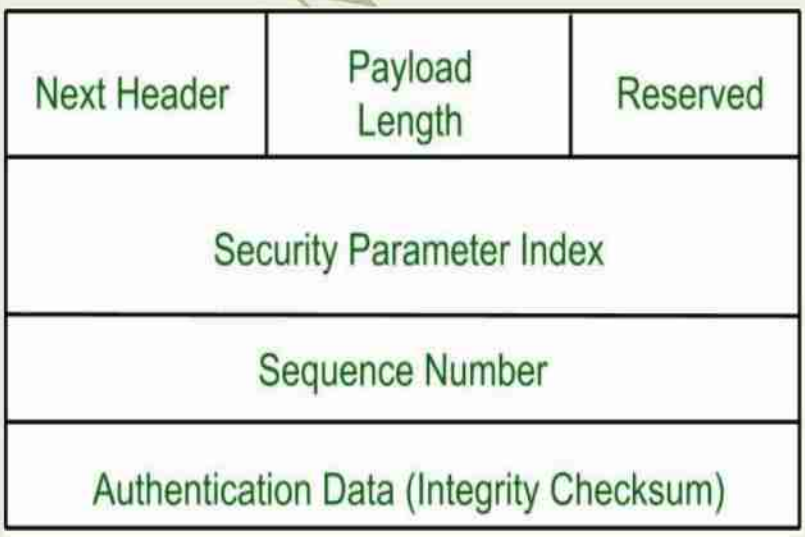


Fig 5.13 AH Protocol

Authentication Header covers the packet format and general issue related to the use of AH forpacket authentication and integrity.

- **Authentication Algorithm**:
  Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.
- **DOI (Domain of Interpretation):**
  DOI is the identifier which support both AH and ESP protocols. It contains values neededfor documentation related to each other.
- **Key Management:**
  Key Management contains the document that describes how the keys are exchanged between sender and receiver.

5.7 Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) is a member of the Internet Protocol Security (IPsec) set of protocols that encrypt and authenticate the packets of data between computers using a Virtual Private Network (VPN). The focus and layer on which ESP operates makes it possible for VPNs to function securely.

Being one of the most popular tools used in network security, Encapsulating Security Payload (abbreviated as ESP) offers the help we need in keeping the integrity, authenticity and confidentiality of the information we send across networks. Keep reading to learn more!

With the technological advancements, the way we conduct our business processes has changed immensely. Now, we heavily rely on the internet technologies and transfer massive amounts of data daily. For this data traffic, we often employ wireless and wired networks. As a result, network security and necessary cybersecurity measures gain importance each day.

Being one of the most popular tools used in network security, Encapsulating Security Payload (abbreviated as ESP) offers the help we need in keeping the integrity, authenticity andconfidentiality of the information we send across networks. In this article, we will take a closer look at what Encapsulating Security Payload is. Keep reading to learn more.

Encapsulating Security Payload

Encapsulating Security Payload (abbr. ESP) is a protocol within the scope of the IPSec.

The information traffic on a network is provided with packets of data. In other words, when you want to send or receive a data through a network, it is turned into packets of information so that it can travel within the network. Similar to the data packages, payload is also sent through the network and it contains the ‘actual’ information, the intended message.

The Encapsulating Security Payload aims to offer necessary security measures for these packets of data and/or payloads. With the help of Encapsulating Security Payload, confidentiality, integrity and authentication of payloads and data packets in IPv4 and IPv6 networks.

Working of Encapsulating Security Payload

Also known as a transport layer security protocol, the Encapsulating Security Payload is able to function with both the IPv6 and IPv4 protocols. The way ESP operates is pretty straightforward: It is inserted between the Internet Protocol/IP header and upper layer protocols such as UDP, ICMP or TCP. In this position, the ESP takes the form of a header.

Usage of Encapsulating Security Payload

Although the Encapsulating Security Payload offers many benefits, it can be applied in only two ways: Tunnel mode and transport mode.

In the tunnel mode, a new IP header is created and used as the outermost IP header. It is followed by the Encapsulating Security Payload Header and original datagram. Tunnel mode is a must for the gateways.

In the transportation mode, the IP header is neither authenticated nor encrypted. As a result, your addressing information can potentially be leaked during the datagram transit. Transport mode often uses less processing, that is why most hosts prefer Encapsulating Security Payload in transport mode.

What are the benefits of the Encapsulating Security Payload?

The Encapsulating Security Payload offers all the functions of the Authentication Header, which are anti-replay protection, authentication and data integrity. On the other hand, the ESPdiffers from the Authentication Header in terms of data confidentiality: the ESP can provide data confidentiality while the Authentication Header cannot.

Moreover, the Encapsulating Security Protocol Payload aims to provide various services including but not limited to:

- Maintaining the confidentiality of datagrams with encryption
- Using security gateways to limit the traffic flow confidentiality
- Authenticating the origin of data using a public key encryption
- Providing antireplay services with the help of the sequence number mechanism given by the Authentication Header

In business environments, we use network technologies very often. They allow us to share resources and files, set communication protocols and such. As much as they streamline and accelerate our business processes, they can also pose a serious vulnerability for our cyber security. An intruder or a hacker can infiltrate into our networks, steal our valuable information or lock us out of our systems. That is why network security is one of the most important practices in cybersecurity.

Most organizations rely on firewalls for their network security needs. A firewall can be defined as a network security system that allows the cybersecurity professionals to monitor and control the network traffic. In other words, a firewall sets the boundary between the internal and external network. There are two main types of firewalls:

- Network-based firewalls: They are often positioned on the LANs, intranets or WANs of the gateway computers.
- Host-based firewalls: They are implemented on the network host itself in order to protect the entire network traffic. Host-based firewalls can be a part of the operating system or an agent application in order to offer an additional layer of security.

Stateful Inspection

The term stateful inspection (also known as the dynamic packet filtering) refers toa distinguished firewall technology. It aims to monitor the active connections on a network. Moreover, the process of stateful inspection determines which network packets should be allowed through the firewall by utilizing the information regarding active connections.

Stateful inspection keeps track of each connection and constantly checks if they are valid. That is why

it offers a better protection than its predecessors.

In a firewall where the stateful inspection is implemented, the network administrator can customize the parameters in order to meet the unique needs of the organization.

Benefit of implementing stateful inspection

Before stateful inspection has become mainstream, similar technology called static packet filtering was in use. This older alternative only checks the headers of the packets in order to determine whether they should be allowed through the firewall. As a result, a hacker can simply indicate “reply” in the header in order to extract information from the network. On thecontrary, stateful inspection aims to carry out a more sophisticated investigation. That is why it analyses the application layer of the packets. A dynamic packet filter like stateful inspection can offer a better security posture for networks through recording the sessioninformation like port numbers or IP addresses.

In other words, stateful inspection is better at keeping the intruders away from your network since it uses a more refined technology.

5.8 Internet key Exchange-

Internet Key Exchange (IKE) is a key management protocol standard used in conjunction with the Internet Protocol Security (IPSec) standard protocol. It provides security for virtual private networks' (VPNs) negotiations and network access to random hosts. It can also be described as a method for exchanging keys for encryption and authentication over an unsecured medium, suchas the Internet.

IKE is a hybrid protocol based on:

- ISAKMP (RFC2408): Internet Security Association and Key Management Protocols are used for negotiation and establishment of security associations. This protocol establishes a secure connection between two IPSec peers.
- Oakley (RFC2412): This protocol is used for key agreement or key exchange. Oakley defines the mechanism that is used for key exchange over an IKE session. The default algorithm for key exchange used by this protocol is the Diffie-Hellman algorithm.
- SKEME: This protocol is another version for key exchange

IKE enhances IPsec by providing additional features along with flexibility. IPsec, however, can be configured without IKE.

IKE has many benefits. It eliminates the need to manually specify all the IPSec security parameters at both peers. It allows the user to specify a particular lifetime for the IPsec security association. Furthermore, encryption can be changed during IPsec sessions. Moreover, it permits certification authority. Finally, it allows dynamic authentication ofpeers.

The IKE works in two steps. The first step establishes an authenticated communication channel between the peers, by using algorithms like the Diffie-Hellman key exchange, which generates a shared key to further encrypt IKE communications. The communication channel formed as a result of the algorithm is a bi-directional channel. The authentication of the channel is achieved by using a shared key, signatures, or public key encryption.

There are two modes of operation for the first step: main mode, which is utilized to protectthe identity of the peers, and aggressive mode, which is used when the security of the identityof the peers is not an important issue. During the second step, the peers use the secure communication channel to set up security negotiations on behalf of other services like IPSec. These negotiation procedures give rise to two unidirectional channels of which one is inbound and the other outbound. The mode of operation for the second step is the Quick mode.

IKE provides three different methods for peer authentication: authentication using a pre- shared secret, authentication using RSA encrypted nonces, and authentication using RSA signatures. IKE uses the HMAC functions to guarantee the integrity of an IKE session. When an IKE session lifetime expires, a new Diffie-Hellman exchange is performed and the IKE SA is re-established.

5.9 WEB SECURITY

Introduction: - World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. The Web presents new challenges not generally appreciated in the context of computer and network security:

The Internet is two ways. Unlike traditional publishing environments, even electronic publishing systems involving Teledex, voice response, or fax-back, the Web is vulnerable to attacks on the Web servers over the Internet.

The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for transactions. Reputations can be damaged.

Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws. The short history of the Web is filled with examples of new and upgraded systems, properly installed, that are vulnerable to a variety of security attacks.

A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems

not part of the Web itself but connected to the server at the local site.

Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

Web Security Threats: One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site. Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server. Issues of server and browser security fall into the category of computer system security.

	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

Fig 5.14: Threats, consequences and countermeasures-web security

5.9 SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSLis not a single protocol but rather two layers of protocols, as illustrated in Figure. The SSL Record Protocol provides basic security services to various higher layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL.

Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols are used in the management of SSL exchange. Two important SSL concepts are the SSL session and theSSL connection, which are defined in the specification as follows. • Connection: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session. • Session: AnSSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic

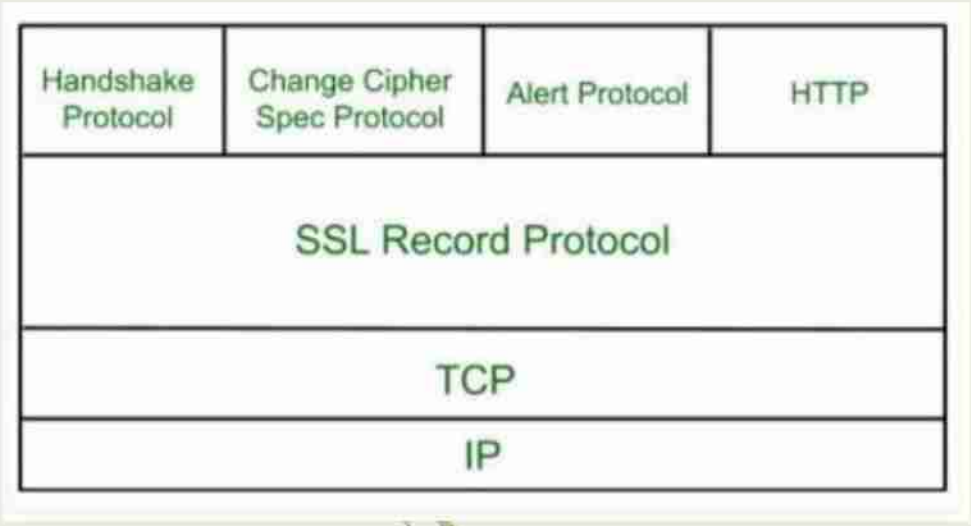


Fig. 5.15 SSL Protocol Stack

security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection. Between anypair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice. There are a number of states associated with each session. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.

A session state is defined by the following parameters.

Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

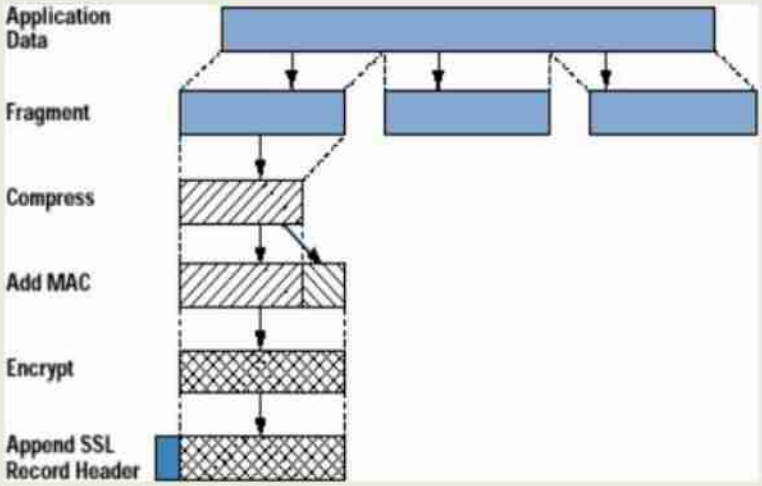


Fig. 5.16 SSL Record Protocol Operation

Peer certificate: An X509.v3 certificate of the peer. This element of the state may be null.

Compression method: The algorithm used to compress data prior to encryption.

Cipher spec: Specifies the bulk data encryption algorithm (such as null,AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.

Master secret: 48-byte secret shared between the client and server.

Is resumable: A flag indicating whether the session can be used to initiate new connections. A connection state is defined by the following parameters.
Server and client random: Byte sequences that are chosen by the server and client for each connection.

Server write MAC secret: The secret key used in MAC operations on data sent by the server.

Client write MAC secret: The secret key used in MAC operations on data sent by the client. • Server write key: The secret encryption key for data encrypted by the server and decrypted by the client.

Client write key: The symmetric encryption key for data encrypted by the client and decrypted by the server.

Initialization vectors: When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key.This field is first initialized by the SSL Handshake Protocol. Thereafter, the final ciphertext block from each record is preserved for use as the IV with the following record.

Sequence numbers: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264 – 1.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

Confidentiality: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.

Message Integrity: The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users. The first step is fragmentation. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), nocompression algorithm is specified, so the default compression algorithm is null. The next step in processing is to compute a message authentication code over the compressed data. Forthis purpose, a shared secret key is used. The calculation is defined as hash(MAC_write_secret || pad_2||

hash(MAC_write_secret || pad_1||seq_num ||

SSLCompressed.type || SSLCompressed.length ||

SSLCompressed.fragment))

where || = concatenation MAC_write_secret =

shared secret key hash = cryptographic hash

algorithm; eitherMD5 or SHA-1

pad_1 = the byte 0x36 (0011 0110) repeated48

times (384 bits) for MD5 and 40

times (320 bits) for

SHA-1 pad_2 = the byte 0x5C (0101 1100) repeated 48
times for MD5 and 40 times for SHA-1

seq_num = the sequence number for this message

SSLCompressed.type = the higher-level protocol used to process this fragment SSLCompressed.length = the length of the compressed fragment

SSLCompressed.fragment = the compressed fragment (if compression is not used, this is the plaintext fragment)

IPSec Protocol:

It is an Internet Engineering Task Force standard suite of protocols between two communication points. It can also be defined as the encrypted, decrypted and authenticated packets. It generally uses cryptographic

security services to protect communications. It can be seen that network-level peer and data origin authentication, data integrity, data encryption, and protection are supported by IPsec.

For Example, IPSec can be used in between two routers in order to create a site-to-site VPN and between a firewall and windows host for a remote access VPN.

SSL:

It is a networking protocol that is used at the transport layer to provide a secure connection between the client and the server over the internet. It is a transparent protocol that requires little interaction from the end-user when establishing a secure session. SSL Tunneling involves a client that requires an SSL connection to a backend service or secure server via a proxy server.

For Example, For securing the communication between a web browser and a web server, he SSL is used.

Difference between IPsec and SSL

IPSec	SSL
Internet protocol security (IPsec) is a set of protocols that provide security for Internet Protocol.	SSL is a secure protocol developed for sending information securely over the Internet.
It Work in Internet Layer of the OSI model.	It Work in Between the transport layer and application layer of the OSI model.
Configuration of IPsec is Complex	Configuration of SSL is Comparatively Simple
IPsec is used to secure a Virtual Private Network.	SSL is used to secure web transactions.
Installation process is Vendor Non-Specific	Installation process is Vendor Specific
Changes are required to OS for implementation. NO Changes are required to application	No changes are required to OS for implementation but Changes are required to application
IPsec resides in operating system space	SSL resides in user space

Fig 5.17: Difference between IPsec and SSL protocol

5.10 INTRUDERS

One of the **most publicized attacks** to security is the intruder, generally referred to as **hacker or cracker**. Three classes of intruders are as follows:

(i) Masquerader – an individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user"s account.

(ii) Misfeasor – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.

(iii) Clandestine user – an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.
The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.
Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However there is no way in advance to know whether an intruder will be benign or malign.
An analysis of previous attack revealed that there were two levels of hackers:

The high levels were sophisticated users with a thorough knowledge of the technology.

The low levels were the „foot soldiers" who merely use the supplied cracking programs with little understanding of how they work.
one of the results of the growing awareness of the intruder problem has been the establishment of several Computer Emergency Response Teams (CERT). These co-operative ventures collect information about system vulnerabilities and disseminate it to systems managers. Unfortunately, hackers can also gain access to CERT reports.
In addition to running password cracking programs, the intruders attempted to modify login software to enable them to capture passwords of users logging onto the systems.

Intrusion techniques
The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruders to acquire information that should be protected. In most cases, the information is in the form of a user password.
Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. The password files can be protected in one of the two ways:

(i) One way encryption – the system stores only an encrypted form of user"s password. In practice, the system usually performs a one way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.
(ii) Access control – access to the password file is limited to one or a very few accounts.

The following techniques are used for learning passwords.

- Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.

- Exhaustively try all short passwords.

- Try words in the system‟s online dictionary or a list of likely passwords.
- Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.
- Try user‟s phone number, social security numbers and room numbers.
- Try all legitimate license plate numbers.
- Use a torjan horse to bypass restriction on access.
- Tap the line between a remote user and the host system.

Two principle countermeasures:

Detection – concerned with learning of an attack, either before or after its success.
Prevention – challenging security goal and an uphill bottle at all times.

INTRUSION DETECTION:

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.
- An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.
- Intrusion detection is based on the assumption that the behaviour of the intruder differs from that of a legitimate user in ways that can be quantified.

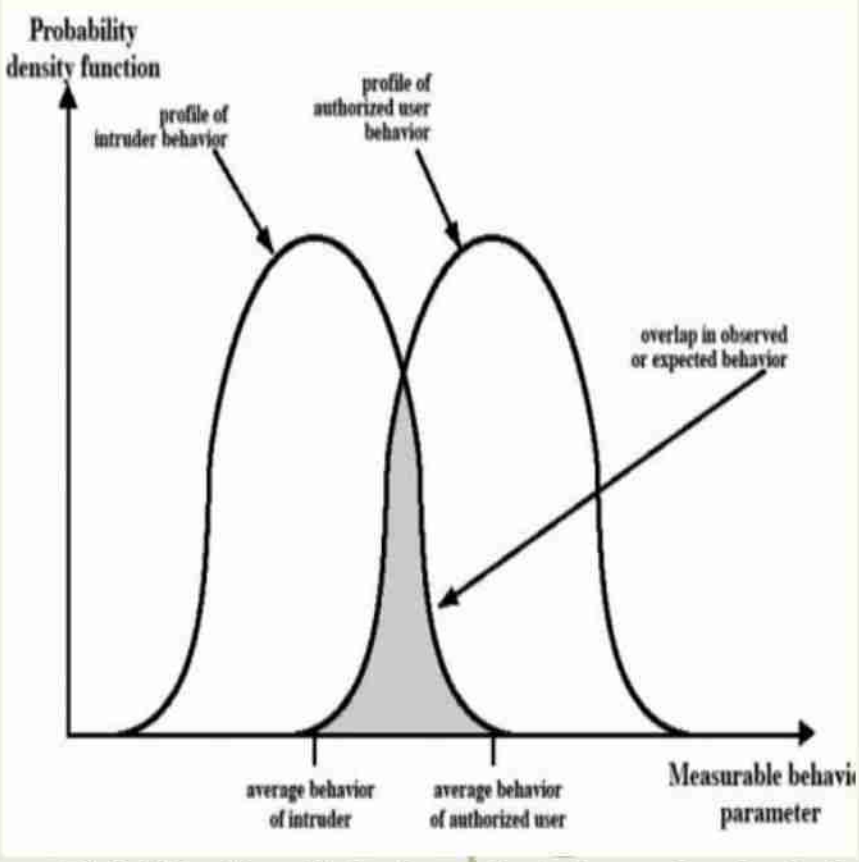


Fig5.18 Profiles of behaviour of intruders and authorized users

Figure suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behaviour of an intruder differs from the typical behaviour of an authorized user, there is an overlap in these behaviours. Thus, a loose interpretation of intruder behaviour, which will catch more intruders, will also lead to a few "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behaviour will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

1. The approaches to intrusion detection:

Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.

Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

Anomaly detection: Rules are developed to detect deviation from previous usage patterns.

Penetration identification: An expert system approach that searches for suspicious behavior.

In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders. On the other hand, such techniques may be unable to deal with misfeasors. For

such attacks, rule-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks.

Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

Native audit records: Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.

Detection-specific audit records: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Each audit record contains the following fields:

- **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users.
- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures

Resource-Usage: A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time).

Time-Stamp: Unique time-and-date stamp identifying when the action took place. Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command

COPY GAME.EXE TO <Library>GAME.EXE

issued by Smith to copy an executable file GAME from the current directory to the <Library> directory. The following audit records may be generated:

Smith	execute	<Library>COPY.EXE	0	CPU = 00002	11058721678
Smith	read	<Smith>GAME.EXE	0	RECORDS = 0	11058721679
Smith	execute	<Library>COPY.EXE	write-viol	RECORDS = 0	11058721680

In this case, the copy is aborted because Smith does not have write permission to <Library>. The decomposition of a user operation into elementary actions has three advantages:

Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus, the system can detect attempted subversions of access

Single-object, single-action audit records simplify the model and the implementation.

Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detection-specific audit records.

1 Statistical Anomaly Detection:

As was mentioned, statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems. **Threshold detection involves** counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed. Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined.

2. Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on several quantitative metrics that can be used to measure user behavior. Examples of metrics that are useful for profile-based intrusion detection are the following:

(a) **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.

(b) **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.

(c) **Interval timer**: The length of time between two related events. An example is the length of time between successive logins to an account.

(d) **Resource utilization**: Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution.

Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits.

- Mean and standard deviation
- Multivariate
- Markov process
- Time series

- Operational

The simplest statistical test is to measure the mean and standard deviation of a parameter over some historical period. This gives a reflection of the average behavior and its variability.
A multivariate model is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time).
A Markov process model is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.
A time series model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing.
Finally, an operational model is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits.

3 Rule-Based Intrusion Detection

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.
Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analysed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behaviour patterns of users, programs, privileges, time slots, terminals, and so on. Current behaviour is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behaviour.
As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behaviour and, in effect, assuming that the future will be like the past
Rule-based penetration identification takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses. Example heuristics are the following:

- Users should not read files in other users' personal directories.
- Users must not write other users' files.
- Users who log in after hours often access the same files they used earlier.
- Users do not generally open disk devices directly but rely on higher-level operating system utilities.
- Users should not be logged in more than once to the same system.
- Users do not make copies of system programs.

2 The Base-Rate Fallacy

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms.

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating.

3 Distributed Intrusion Detection

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN Porras points out the following major issues in the design of a distributed intrusion detection system

A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.

One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data.

Either a centralized or decentralized architecture can be used.

Below figure shows the overall architecture, which consists of three main components:

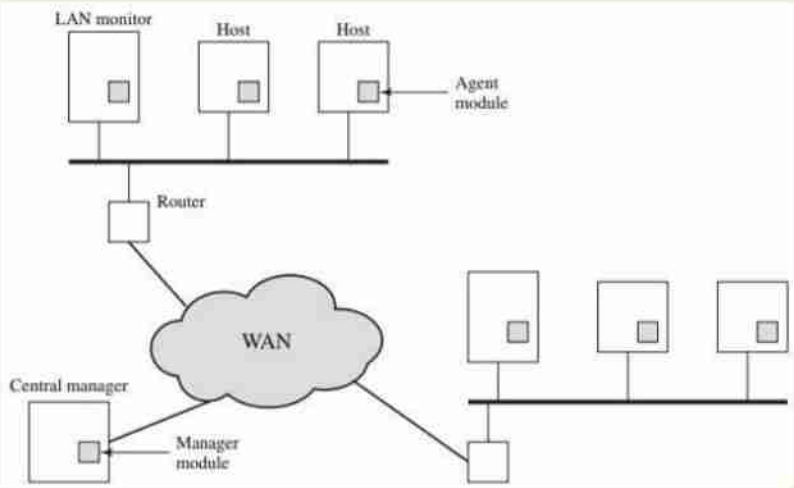


Fig 5.19: Architecture for Distributed Instrusion detection

Host agent module: An audit collection module operating as a background process on

a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.

LAN monitor agent module: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.

Central manager module: Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

- The scheme is designed to be independent of any operating system or system auditing implementation.
- The agent captures each audit record produced by the native audit collection system.
- A filter is applied that retains only those records that are of security interest.
- These records are then reformatted into a standardized format referred to as the host audit record (HAR).
- Next, a template-driven logic module analyses the records for suspicious activity. At the lowest level, the agent scans for notable events that are of interest independent of any past events.
- Examples include failed file accesses, accessing system files, and changing a file's access control.
- At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures).
- Finally, the agent looks for anomalous behaviour of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.
- When suspicious activity is detected, an alert is sent to the central manager.
- The central manager includes an expert system that can draw inferences from received data.
- The manager may also query individual systems for copies of HARs to correlate with those from other agents.
- The LAN monitor agent also supplies information to the central manager.
- The LAN monitor agent audits host-host connections, services used, and volume of traffic.
- It searches for significant events, such as sudden changes in network load, the use of
- security-related services, and network activities such as rlogin.

The architecture is quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from several sites and networks to detect suspicious activity that would otherwise remain undetected.

4 Honeypots

A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots are decoy systems that are designed to lure a potential attacker away from critical systems.

Honeypots are designed to

- divert an attacker from accessing critical systems
- collect information about the attacker's activity
- encourage the attacker to stay on the system long enough for administrators to respond

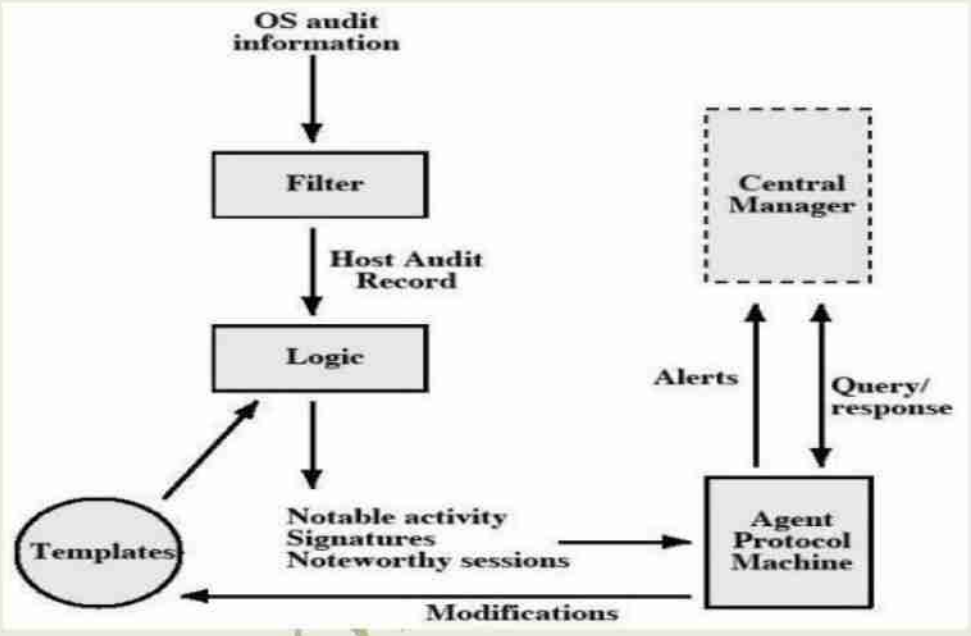


Fig 5.20 Honeypots

These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect.

5 Intrusion Detection Exchange Format

To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group.

The outputs of this working group include the following:

a. A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements.

b. A common intrusion language specification, which describes data formats that satisfy the requirements.

c. A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised data formats relate to them.

PASSWORD MANAGEMENT

1. Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

- The ID determines whether the user is authorized to gain access to a system.
- The ID determines the privileges accorded to the user.
- The ID is used in ,what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

2. The Vulnerability of Passwords

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, the following procedure is employed. Each user selects a password of up to eight printable characters in length. This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. The encryption routine, known as crypt(3), is based on DES. The DES algorithm is modified using a 12-bit "salt" value. Typically, this value is related to the time at which the password is assigned to the user. The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros.The output of the algorithm then serves as input for a second encryption. This process is repeated for a total of 25 encryptions. The resulting 64-bit output is then translated into an 11-character sequence. The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID. This method has been shown to be secure against a variety of cryptanalytic attacks

The salt serves three purposes:

(i) It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.

(ii) It effectively increases the length of the password without requiring the user to remember two additional characters.

(iii) It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.

When a user attempts to log on to a UNIX system, the user provides an ID and a password. The

operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password. The salt and user-supplied password are used as input to the encryption routine. If the result matches the stored value, the password is accepted.The encryption routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25.

Thus, there are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine.

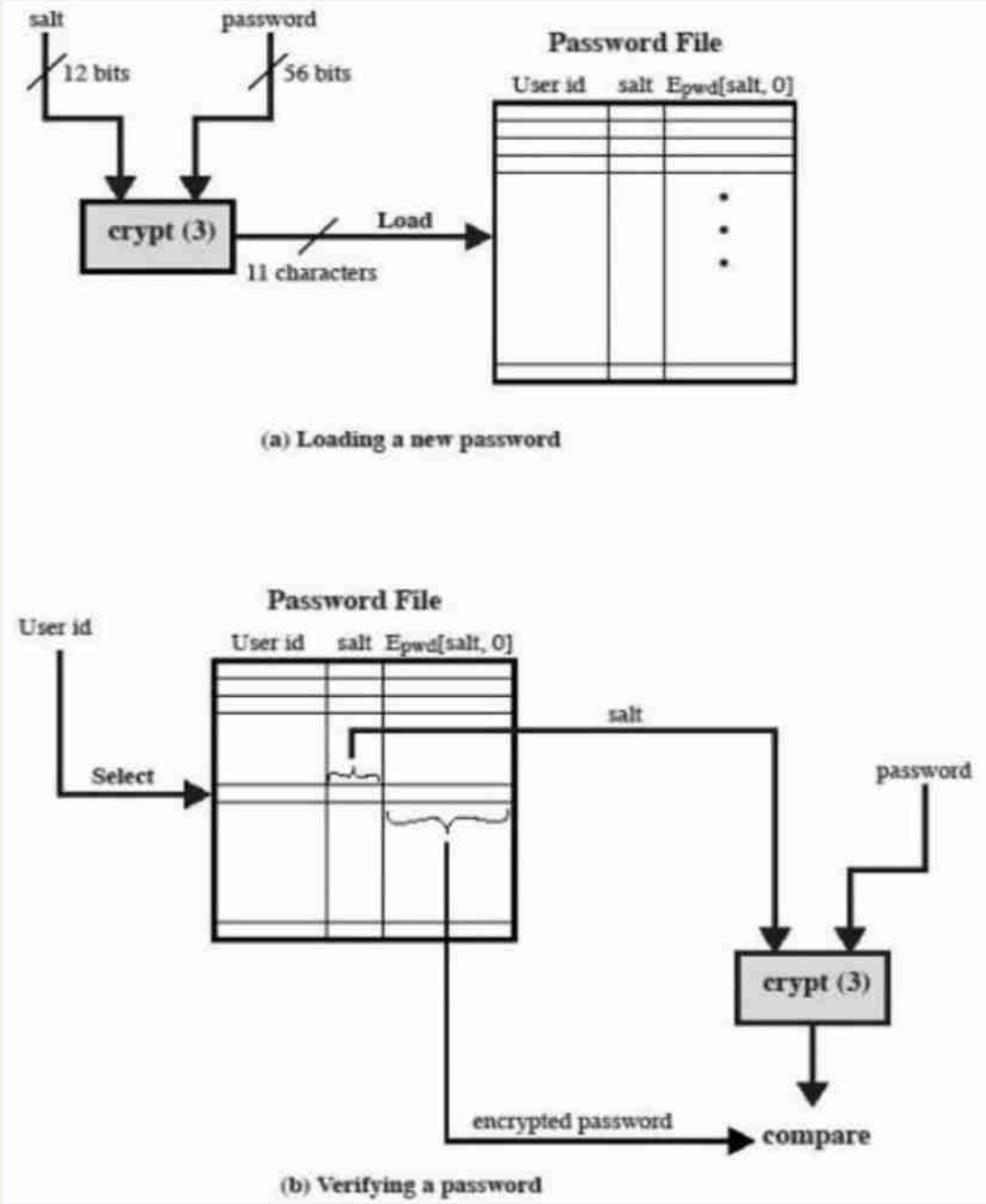


Fig 5.21 UNIX password scheme

As an example**, a password cracker was reported on the Internet in** August 1993. Using a Thinking Machines Corporation parallel computer, a performance of 1560 encryptions per second per vector unit was achieved. With four vector units per processing node (a standard configuration), this works out to 800,000 encryptions per second on a 128-node machine (which is a modest size) and 6.4 million encryptions per second on a 1024-node machine.
Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward.
Following strategy was used:

- Try the user's name, initials, account name, and other relevant personal information. In all, 130 different permutations for each user were tried.
- Try words from various dictionaries.
- Try various permutations on the words from step 2.
- Try various capitalization permutations on the words from step 2 that were not considered in step 3. This added almost 2 million additional words to the list.

3. Access Control

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user.

Password Selection Strategies
Four basic techniques are in use:

(i) User education
(ii) Computer-generated passwords
(iii)Reactive password checking
(iv)Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines
Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down
A reactive password checking strategy is one in which the system periodically runs its own password cracker to find guessable passwords.
The most promising approach to improved password security is a **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such

checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a large password space that are not likely to be guessed in a dictionary attack.

The first approach is a simple system for rule enforcement. For example, the following rules could be enforced:

- All passwords must be at least eight characters long.
- In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks. These rules could be coupled with advice to the user. Although this approach is superior to simply educating users, it may not be sufficient to thwart password crackers. This scheme alerts crackers as to which passwords not to try but may still make it possible to do password cracking.

Another possible procedure is simply to compile a large dictionary of possible "bad" passwords. When a user selects a password, the system checks to make sure that it is not on the disapproved list.

There are two problems with this approach:

- Space: The dictionary must be very large to be effective.
- Time: The time required to search a large dictionary may itself be large

Two techniques for developing an effective and efficient proactive password checker that is based on rejecting words on a list show promise. One of these develops a Markov model for the generation of guessable passwords. This model shows a language consisting of an alphabet of three characters. The state of the system at any time is the identity of the most recent letter. The value on the transition from one state to another represents the probability that one letter follows another. Thus, the probability that the next letter is b, given that the current letter is a, is 0.5.

In general, a Markov model is a quadruple [m, A, T, k], where m is the number of states in the model, A is the state space, T is the matrix of transition probabilities, and k is the order of the model. For a kth-order model, the probability of making a transition to a particular letter depends on the previous k letters that have been generated.

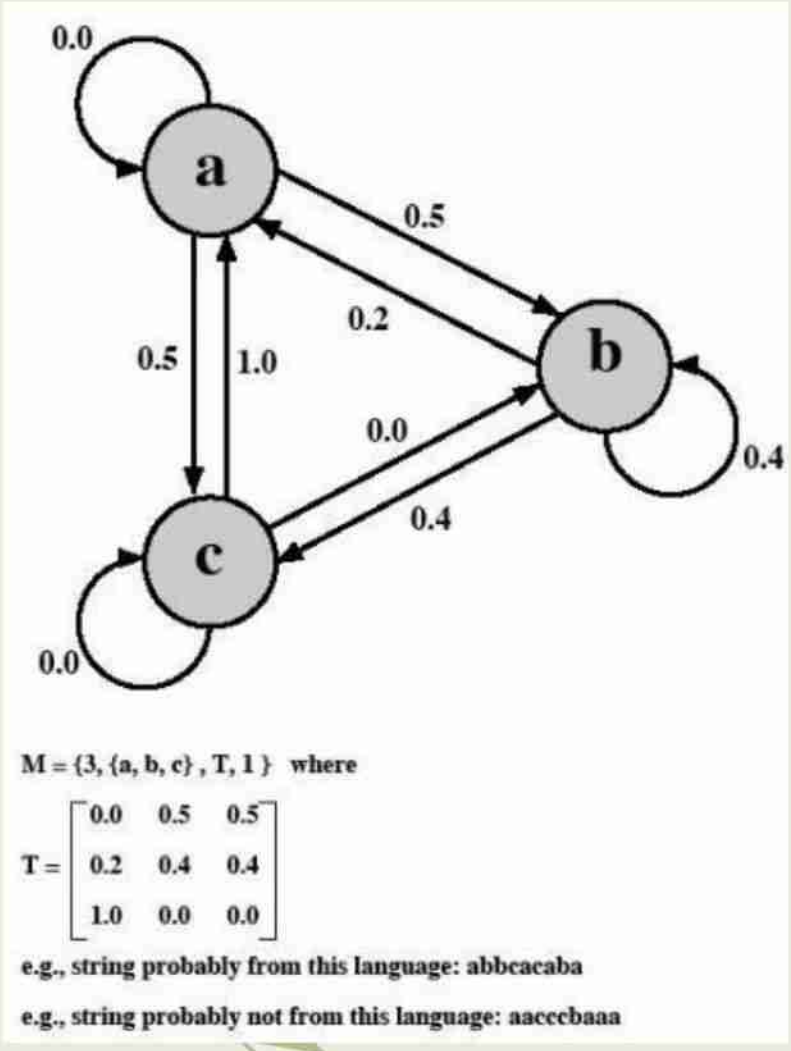


Fig 5.22 second order model

The authors report on the development and use of a second-order model. To begin, a dictionary of guessable passwords is constructed. Then the transition matrix is calculated as follows:

1. Determine the frequency matrix f, where f(i, j, k) is the number of occurrences of the trigram consisting of the ith, jth, and kth character. For example, the password parsnips yields the trigrams par, ars, rsn, sni, nip, and ips.
2. For each bigram ij, calculate f(i, j,∞) as the total number of trigrams beginning with ij. For example, f(a, b,∞) would be the total number of trigrams of the form aba, abb, abc, and so on.
3. Compute the entries of T as follows:

T(i,j,k) = f(i, j, k) / f(i, j,∞)

The result is a model that reflects the structure of the words in the dictionary.

A quite different approach has been reported by Spafford. It is based on the use of a Bloom filter. To begin, we explain the operation of the Bloom filter. A Bloom filter of order k consists of a set of k independent hash functions $H_1(x)$, $H_2(x)$,..., $H_k(x)$, where each function maps a password into a hash value in the range 0 to N - 1

That is,

$$H_i(X_j) = y \ 1 \leq i \leq k; \ 1 \leq j \leq D; \ 0 \leq y \leq N-1$$

where
X_j = jth word in password dictionary
D = number of words in password dictionary

The following procedure is then applied to the dictionary:

- A hash table of N bits is defined, with all bits initially set to 0.
- For each password, its k hash values are calculated, and the corresponding bits in the hash table are set to 1. Thus, if $H_i(X_j) = 67$ for some (i, j), then the sixty-seventh bit of the hash table is set to 1; if the bit already has the value 1, it remains at 1.

When a new password is presented to the checker, its k hash values are calculated. If all the corresponding bits of the hash table are equal to 1, then the password is rejected.

5.11 FIREWALLS

1. Firewall design principles

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. The alternative, increasingly accepted, is the firewall.

The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

2. Firewall characteristics:

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

- Various types of firewalls are used, which implement various types of security policies.

- The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site"s security policy is as follows:

1. Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.
2. Direction control – determines the direction in which particular service request may be initiated and allowed to flow through the firewall.
3. User control – controls access to a service according to which user is attempting to access it.
4. Behaviour control – controls how particular services are used.

Capabilities of firewall

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

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

- A firewall is a convenient platform for several internet functions that are not security related.

- A firewall can serve as the platform for IPsec.

3. Limitations of firewall

The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.

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

· The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to

scan all incoming files, e-mail, and messages for viruses.

4 Types of firewalls

There are 3 common types of firewalls.

(i) Packet filters
(ii) Application-level gateways
(iii)Circuit-level gateways

(i)Packet filtering router

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

Source IP address – IP address of the system that originated the IP packet. Destination IP address – IP address of the system, the IP is trying to reach. Source and destination transport level address – transport level port number. IP protocol field – defines the transport protocol. Interface – for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.

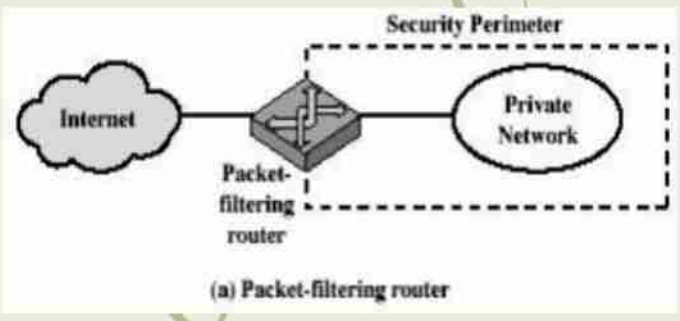


Fig 5.23: Packet filtering router

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken.

Two default policies are possible:

· Default = discard: That which is not expressly permitted is prohibited.
· Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

5. Advantages of packet filter router

- Simple

- Transparent to users
- Very fast

Weakness of packet filter firewalls

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.
- It does not support advanced user authentication schemes.
- They are generally vulnerable to attacks such as layer address spoofing.

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

- IP address spoofing – the intruders transmit packets from the outside with a source IP address field containing an address of an internal host.
- Countermeasure: to discard packet with an inside source address if the packet arrives on an external interface.
- Source routing attacks – the source station specifies the route that a packet should take as it crosses the internet; i.e., it will bypass the firewall.
- Tiny fragment attacks – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

Countermeasure: to discard all packets where the protocol type is TCP and the IP fragment offset is equal to 1.

(ii) Application-level gateway

An application-level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application-level gateways tend to be more secure than packet filters. It is easy to log and audit

all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.

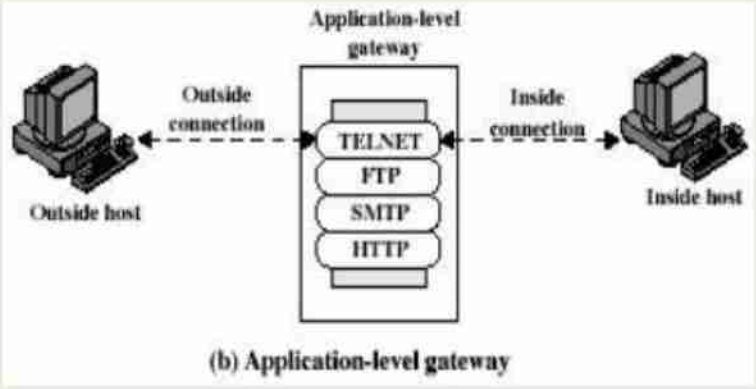


Fig 5.24: Application level gateway

(iii) Circuit level gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.

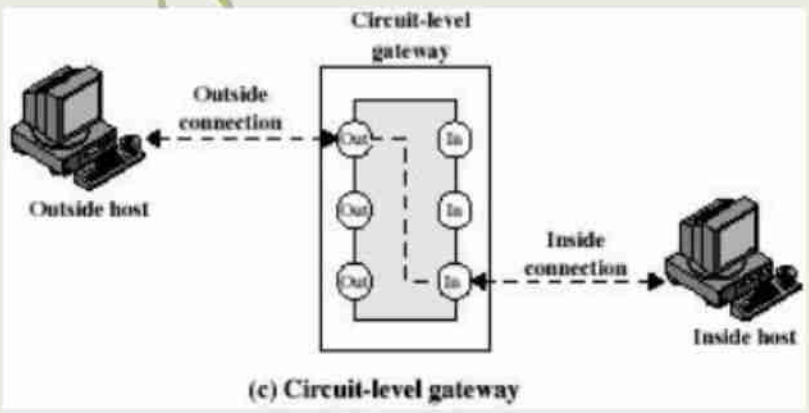


Fig 5.25: Circuit level gateway

Basiton host

It is a system identified by the firewall administrator as a critical strong point in the network's security. The Bastion host serves as a platform for an application level and circuit level gateway.

Common characteristics of a Basiton host are as follows:

- The Bastion host hardware platform executes a secure version of its operating system, making it a trusted system.
- Only the services that the network administrator considers essential are installed on the Bastion host.

- It may require additional authentication before a user is allowed access to the proxy services.

- Each proxy is configured to support only a subset of standard application"s command set.
- Each proxy is configured to allow access only to specific host systems.

- Each proxy maintains detailed audit information by logging all traffic, each connection
- and the duration of each connection.

- Each proxy is independent of other proxies on the Bastion host.

- A proxy generally performs no disk access other than to read its initial configuration file.

- Each proxy runs on a non privileged user in a private and secured directory on the Bastion host.

Firewall configurations

There are 3 common firewall configurations.

1. Screened host firewall, single-homed basiton configuration
2. Screened host firewall, dual homed basiton configuration
3. Screened subnet firewall configuration

1. Screened host firewall, single-homed basiton configuration

In this configuration, the firewall consists of two systems: a packet filtering router and a bastion host. Typically, the router is configured so that

- for traffic from the internet, only IP packets destined for the basiton host are allowed in.

- for traffic from the internal network, only IP packets from the basiton host are allowed out.

The basiton host performs authentication and proxy functions. This configuration has greater security than simply a packet filtering router or an application-level

gateway alone, for two reasons:

· This configuration implements both packet level and application-level filtering, allowing for considerable flexibility in defining security policy.

An intruder must generally penetrate two separate systems before the security of the internal network is compromised.

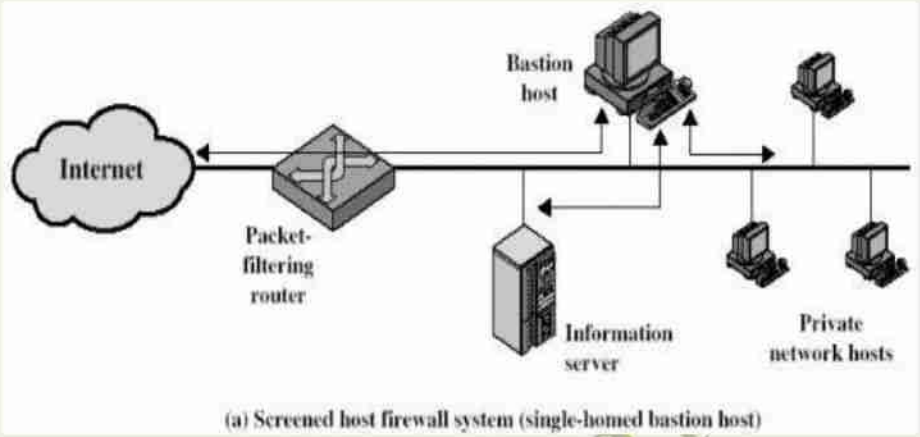


Fig 5.26: Screened host firewall system(single home bastion host)

2. Screened host firewall, dual homed basiton configuration

In the previous configuration, if the packet filtering router is compromised, traffic could flow directly through the router between the internet and the other hosts on the private network. This configuration physically prevents such a security break.

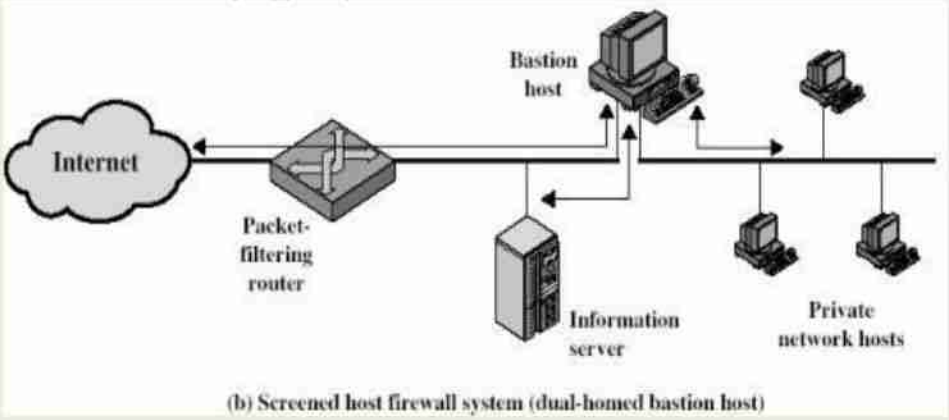


Fig 5.27: Screened host firewall system(dual home bastion host)

3. Screened subnet firewall configuration

In this configuration, two packet filtering routers are used, one between the basiton host and internet and one between the basiton host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the basiton host but may also include one or more information servers and modems for dial-in capability. Typically both the internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages.

There are now three levels of defense to thwart intruders.
The outside router advertises only the existence of the screened subnet to the internet; therefore, the internal network is invisible to the internet.
Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the internal network cannot construct direct routes to the internet.

5.12 MALICIOUS SOFTWARE

MALICIOUS SOFTWARE

Types Of Malicious Software

- **Backdoor**
- **Logic Bomb**
- **Trojan Horses**
- **Mobile Code**
- **Multiple-Threat Malware**

Viruses

- **The Nature of Viruses**
- **Viruses Classification**
- **Virus Kits**
- **Macro Viruses**
- **E-Mail Viruses**

Virus Countermeasures

- **Antivirus Approaches**
- **Advanced Antivirus Techniques**

Worms

- **The Morris Worm**
- **Worm Propagation**
- **Model Recent Worm Attacks**
- **State of Worm Technology**
- **Mobile Phone**
- **Worms Worm Countermeasures**

Distributed Denial Of Service Attacks

- **DDoS Attack Description**
- **Constructing the Attack Network**
- **DDoS Countermeasures**

KEY POINTS

◆ Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.

◆ A virus is a piece of software that can “infect” other programs by modify- ing them; the modification includes a copy of the virus program, which can then go on to infect other programs.

◆ A worm is a program that can replicate itself and send copies from com- puter to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. In addition to propaga- tion, the worm usually performs some unwanted function.

◆ A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from

using that service.

- A distributed denial of service attack is launched from multiple coordinated sources.

(i) VIRUSES AND RELATED THREATS

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

1. Malicious Programs

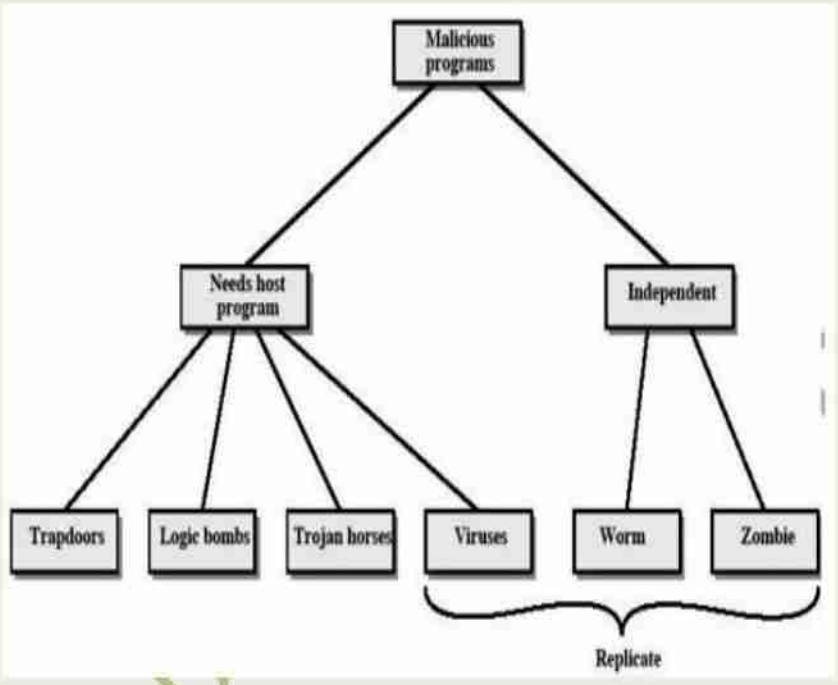


Fig 5.28 Taxonomy of Malicious Programs

Malicious software can be divided into two categories: those that need a host program, and those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

Name	Description
Virus	Attaches itself to a program and propagates copies of itself to other programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor (trapdoor)	Program modification that allows unauthorized access to functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely
Kit (virus generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Keyloggers	Captures keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

(ii) **The Nature of Viruses**

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

A virus can do anything that other programs do. The only difference is that it attaches itself to

another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

- Dormant phase: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

- Propagation phase: The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

- Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

- Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

An infected program begins with the virus code and works as follows.

The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.
A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length.. The key lines in this virus are numbered. We assume that program P_1 is infected with the virus CV. When this program is invoked, control passes to its virus,

which performs the following steps:

1. For each uninfected file P_2 that is found, the virus first compresses that file to produce P'_2, which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P'_1, is uncompressed.
4. The uncompressed original program is executed.

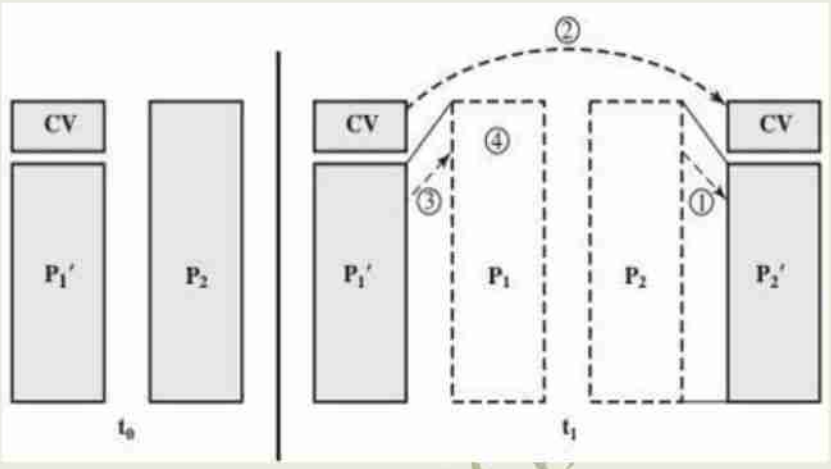


Fig 5.29 A Compression Virus

In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

4. Initial Infection

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of iron and write all one's own system and application programs, one is vulnerable.

Types of Viruses

following categories as being among the most significant types of viruses:

Parasitic virus: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.

Memory-resident virus: Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.

Boot sector virus: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

Stealth virus: A form of virus explicitly designed to hide itself from detection by antivirus software.

Polymorphic virus: A virus that mutates with every infection, making detection by the "signature" of the virus impossible.

Metamorphic virus: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration,

increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.

One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns.

Macro Viruses

In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:

A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.

Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.

Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input.

Successive releases of Word provide increased protection against macro viruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros. Various antivirus product vendors have also developed tools to detect and correct macro viruses.

E-mail Viruses

A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.

The virus does local damage.

Worms

A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again. Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

To replicate itself, a network worm uses some sort of network vehicle. Examples include the following:

- Electronic mail facility: A worm mails a copy of itself to other systems.
- Remote execution capability: A worm executes a copy of itself on another system.
- Remote login capability: A worm logs onto a remote system as a user and then uses

commands to copy itself from one system to the other.
The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.
A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally performs the following functions:

1. Search for other systems to infect by examining host tables or similar repositories of remote system addresses.

2. Establish a connection with a remote system.

3. Copy itself to the remote system and cause the copy to be run.

As with viruses, network worms are difficult to counter.

The Morris Worm

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password-cracking program that tried
Each user's account name and simple permutations of it
A list of 432 built-in passwords that Morris thought to be likely candidates All the words in the local system directory

- It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.
- It exploited a trapdoor in the debug option of the remote process that receives and sends mail.
- If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter.

Recent Worm Attacks

In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms:

- from client to client via e-mail
- from client to client via open network shares
- from Web server to client via browsing of compromised Web sites
- from client to Web server via active scanning for and exploitation of various Microsoft IIS 4.0 / 5.0 directory traversal vulnerabilities
- from client to Web server via scanning for the back doors left behind by the "Code Red II" worms

The worm modifies Web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects and creates numerous copies of itself under various filenames.

In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server.
Mydoom is a mass-mailing e-mail worm that appeared in 2004
Network Virus Countermeasures

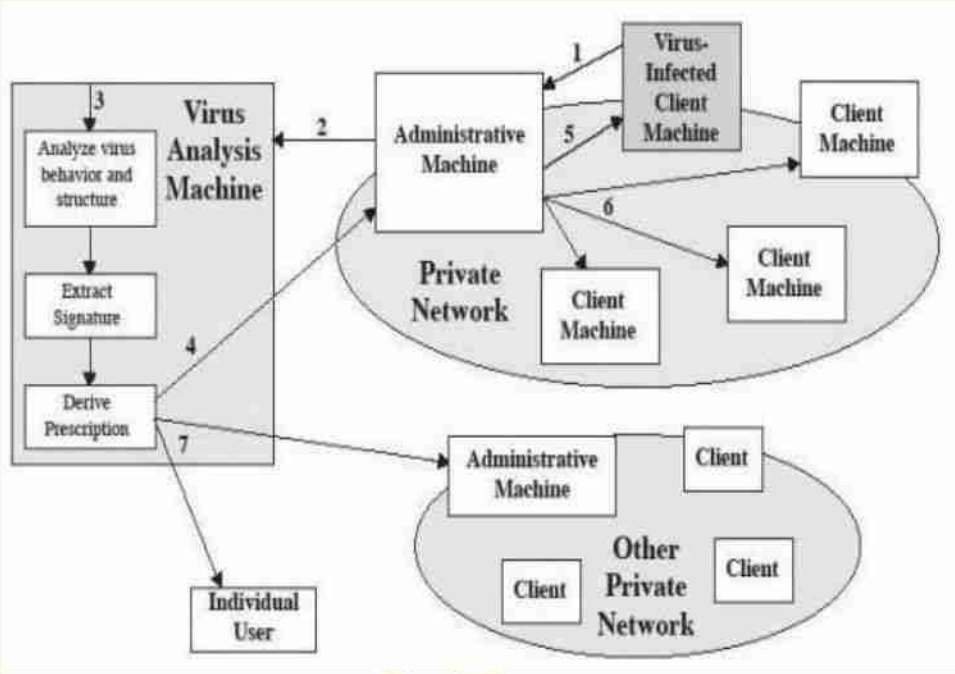


Fig 5.30: Network Virus Countermeasures

The ideal solution to the threat of viruses is prevention: The next best approach is to be able to do the following:

VIRUS COUNTERMEASURES

Antivirus Approaches

The ideal solution to the threat of viruses is prevention: The next best approach is to be able to do the following:

Detection: Once the infection has occurred, determine that it has occurred and locate the virus.

Identification: Once detection has been achieved, identify the specific virus that has infected a program.

Removal: Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.

There are four generations of antivirus software:

(i) First generation: simple scanners

(ii) Second generation: heuristic scanners

(iii) Third generation: activity traps

(iv) Fourth generation: full-featured protection

A first-generation scanner requires a virus signature to identify a virus.. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the virus cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the virus is prevented from adjusting the program to produce the same hash code as before.

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

Fourth-generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures.

Advanced Antivirus Techniques

More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight two of the most important.

Generic Decryption

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds . In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

CPU emulator: A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

Virus signature scanner: A module that scans the target code looking for known virus signatures.

Emulation control module: Controls the execution of the target code.

Digital Immune System

The digital immune system is a comprehensive approach to virus protection developed by IBM]. The motivation for this development has been the rising threat of Internet-based virus propagation. Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years: Integrated mail systems: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

Mobile-program systems: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

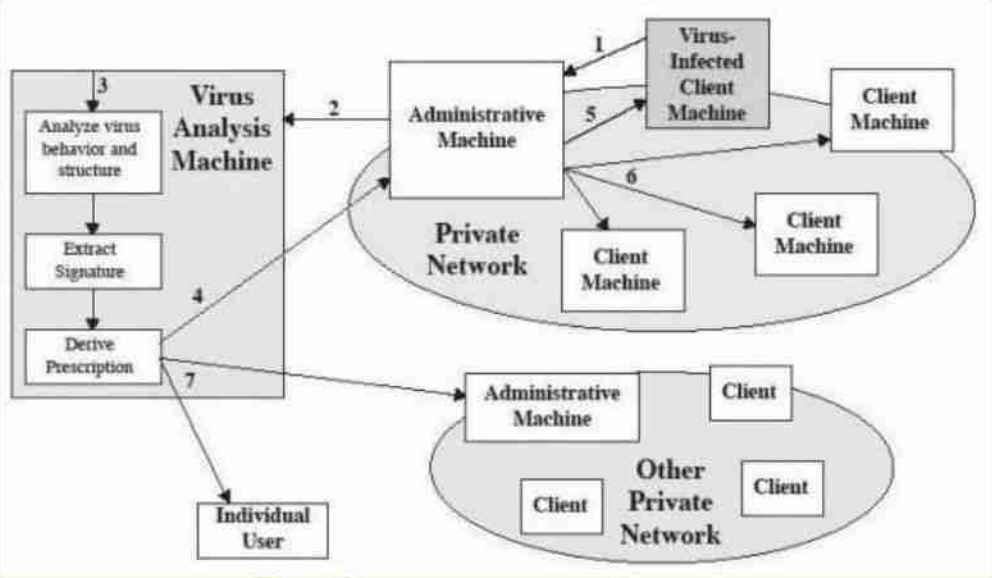


Fig 5.31: Mobile program system

A monitoring program on each PC uses a variety of heuristics based on system behaviour, suspicious changes to programs, or family signature to infer that a virus may be present. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.

The administrative machine encrypts the sample and sends it to a central virus analysis machine. This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored. The virus analysis machine then produces a prescription for identifying and removing the virus.

- The resulting prescription is sent back to the administrative machine.
- The administrative machine forwards the prescription to the infected client.
- The prescription is also forwarded to other clients in the organization.

- Subscribers around the world receive regular antivirus updates that protect them from the new virus.

The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains. By constantly analyzing and monitoring the viruses found in the wild, it should be possible to continually update the digital immune software to keep up with the threat.

Behavior-Blocking Software

Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. Monitored behaviors can include the following:

- Attempts to open, view, delete, and/or modify files;
- Attempts to format disk drives and other unrecoverable disk operations;
- Modifications to the logic of executable files or macros;
- Modification of critical system settings, such as start-up settings;
- Scripting of e-mail and instant messaging clients to send executable content; and
- Initiation of network communications.

If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established antivirus detection techniques as fingerprinting or heuristics.

DISTRIBUTED DENIAL OF SERVICE ATTACKS

A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or net- work node, then it is simply referred to as a DoS attack. A more serious threat is posed by a DDoS attack. In a DDoS attack, an attacker can recruit several hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target. This section is concerned with DDoS att acks. First, we look at the nature and types of attacks. Next, we examine means by which an attacker can recruit a network of hosts for attack launch. Finally, this section looks at countermeasures.

5.13 DDoS Attack Description

A DDoS attack attempts to consume the target's resources so that it cannot provideservice. One way to classify DDoS attacks is in terms of the type of resource that is consumed. Broadly speaking, the resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target isattacked.

A simple example of an **internal resource attack** is the SYN flood attack.

Figure shows the steps involved:

1. The attacker takes control of multiple hosts over the Internet, instructing them to contact the target Webserver.
2. The slave hosts begin sending TCP/IP SYN (synchronize/initialization) packets, with erroneous return IP address information, to the target.
3. Each SYN packet is a request to open a TCP connection. For each such packet, the Web server responds with a SYN/ACK (synchronize/acknowl- edge) packet, trying to establish a TCP connection with a TC P entity at a spu- rious IP address. The Web server maintains a data structure for each SYN request waiting for a response back and becomes bogged down as more t raffic floods in. The result is that legitimate connections are denied while the victim machine is waiting to complete bogus "half-open" connections.

The TCP state data structure is a popular internal resource target but by no means the only one. [CERT01] gives the following examples:

- In many systems, a limited number of data structures are available to hold processinformation(processidentifiers, processtableentries, processslots, etc.). A n intruder may be able to consume these data structures by writing a simple prog ram or script that does nothing but repeatedly create copies of itself.
- An intruder may also attempt to consume disk space in other ways, including generating excessive numbers of mail message intentionally generating errors that must be logged.
- placing files in anonymous ftp areas or network-shared areas

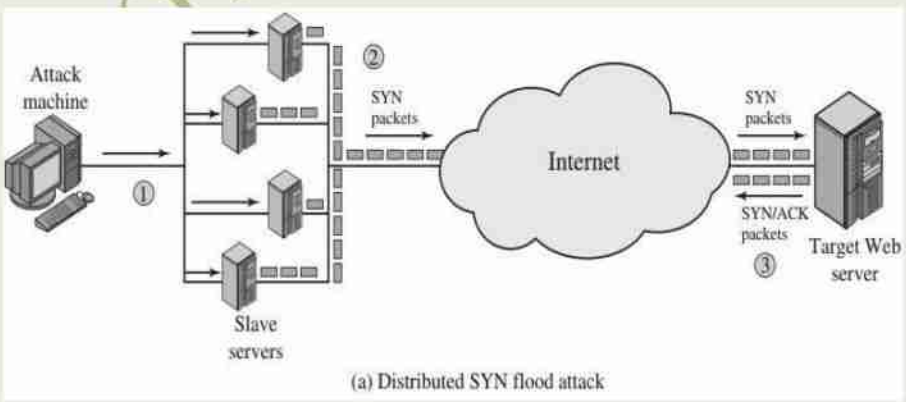


Fig 5.32 Examples of simple DDoS attacks

Figure illustrates an example of an **attack that consumes data transmis- sion resources**. The following steps are involved:

1. The attacker takes control of multiple hosts over the Internet, instructing them to send ICMP ECHO packets3 with the target's spoofed IP address to a group of hosts that act as reflectors, as described subsequently.

2. Nodes at the bounce site receive multiple spoofed requests and respond by send- ing echo reply packets to the target site.

3. The target's router is flooded with packets from the bounce site, leaving no data transmission capacity for legitimate traffic.

Another way to classify DDoS attacks is as either direct or reflector DDoS attacks. In a **direct DDoS** attack (Figure 21.10a), the attacker can implant zom- bie software on a number of sites distributed throughout the Internet. Often, the DDoS attack involvestwo levels ofzombie machines: master zombiesandslavezom- bies. The hosts of both machines have been infected with malicious code. The attacker coordinates and triggers the master zombies, which in turn coordinate and trigger the slave zombies. Theuse oftwo levels ofzombies makes it more difficult to trace the attack back to its source and provides for a more resilient network of attackers.

A **reflector DDoS** attack adds another layer of machines (Figure 21.10b). In this type of attack, the slave zombies construct packets requiring a response that contains the target's IP address as the source IP address in the packet's IP header. These packets are sent to uninfected machines known as reflectors. The uninfected machines respond with packets directed at the target machine. A reflector DDoS attack can easily involve more machines and more traffic than a direct DDoS attack and hence be more damaging. Further, tracing back the attack or filtering out the attack packets is more difficult because the attack comes from widely dispersed uninfected machines.

Constructing the Attack Network

The first step in a DDoS attack is for the attacker to infect a number of machines with zombie software that will ultimately be used to carry out the attack. The essential ingredients in this phase of the attack are the following:

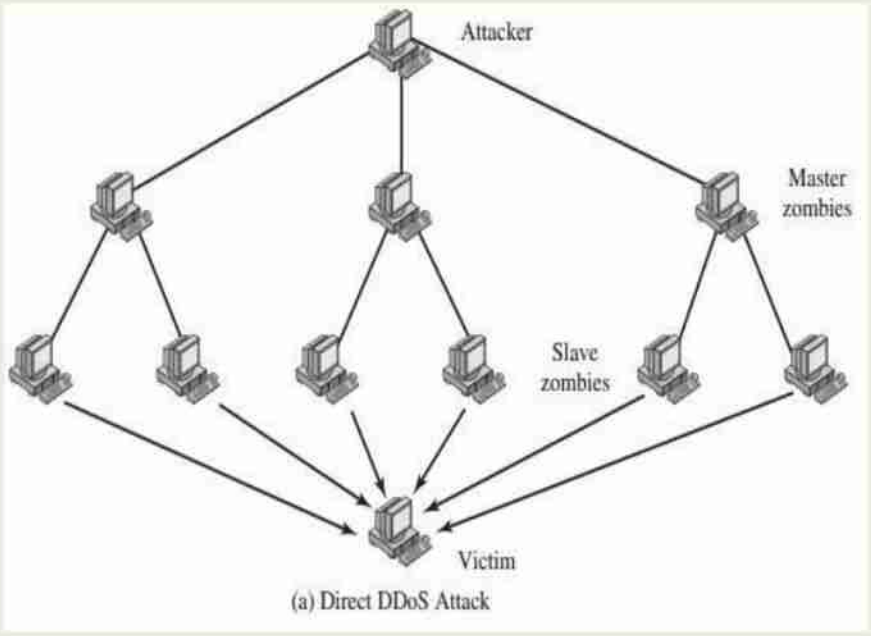
1. Software that can carry out the DDoS attack. The software must be able to run on many machines, must be able to conceal its existence, must be able to communicate with the attacker or have some sort of time-triggered mechanism, and must be able to launch the intended attack toward the target.

2. A vulnerability in a large number of systems. The attacker must become aware of a vulnerability that many system administrators and individual users have failed to patch and that enables the attacker to install the zombie software.

3. A strategy for locating vulnerable machines, a process known as scanning.

In the scanning process, the attacker first seeks out several vulnerable machines and infects them. Then, typically, the zombie software that is installed in the infected machines repeats the same scanning process, until a large distributed network of infected machines is created. [MIRK04] lists the following types of scanning strategies:

Random: Each compromised host probes random addresses in the IP address space, using a different seed. This technique produces a high volume of Internet traffic, which may cause generalized disruption even before the actual attack is launched.



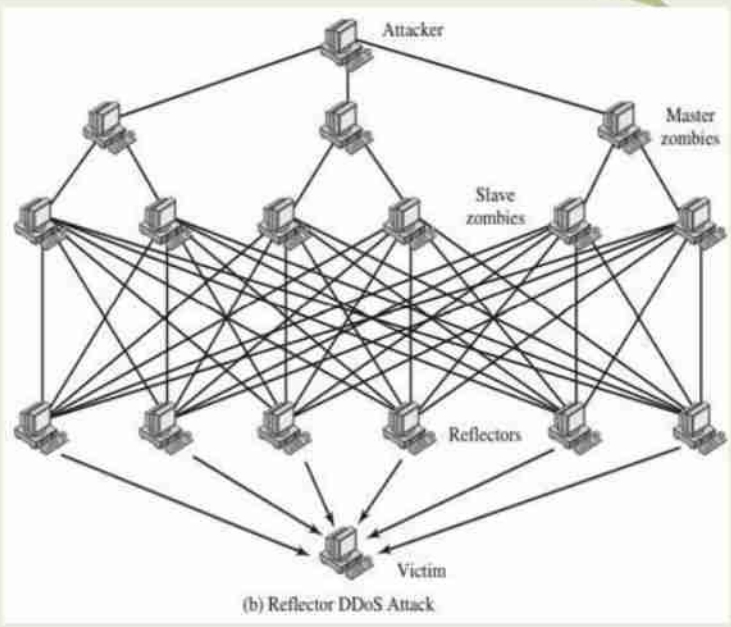


Fig 5.33 Types of flooding-based DDoS attacks

• Hit-List: The attacker first compiles a long list of potential vulnerable machines. This can be a slow process done over a long period to avoid detection that an attack is underway. Once the list is compiled, the attacker begins infecting machines on the list. Each infected machine is provided with a portion of the list to scan. This strategy results in a very short scanning period, which may make it difficult to detect that infection is taking place.

• Topological: This method uses information contained on an infected victim machine to find more hosts to scan.

• Local subnet: If a host can be infected behind a firewall, that host then looks for targets in its own local network. The host uses the subnet address structure to find other hosts that would otherwise be protected by the firewall.

DDoS Countermeasures

In general, there are three lines of defense against DDoS attacks:

• Attack prevention and pre-emption (before the attack): These mechanisms enable the victim to endure attack attempts without denying service to legiti- mate clients. Techniques include enforcing policies for resource consumption and providing
backup resources available on demand. In addition, prevention mechanisms modify
systems and protocols on the Internet to reduce the possi- bility of DDoS attacks.
• Attack detection and filtering (during the attack): These mechanisms attempt to detect the attack as it begins and respond immediately. This minimizes the impact of the attack on the target. Detection involves looking for suspicious patterns of behavior. Response involves filtering out packets likely to be part of the attack.
• Attack source traceback and identification (during and after the attack): This is an attempt to identify the source of the attack as a first step in preventing future attacks. However, this method typically does not yield results fast enough, if at all, to mitigate an ongoing attack.

REFERENCES

1. Willliam Stallings," Network System Essentials "-4th Edition Copyright © 2011Pearson education, Inc., publishing as [Prentice Hall,
2. Atul Khahate, "Cryptography and network security",3rd Edition, Copyright © 2013TMH Publishing
3. Kuldeep Singh Kohar", Network Security", revised reprint 2011.Vayu Education ofIndia, New Delhi.Hall, 1983.